# Mobile Device Security

Dietrich Lehr

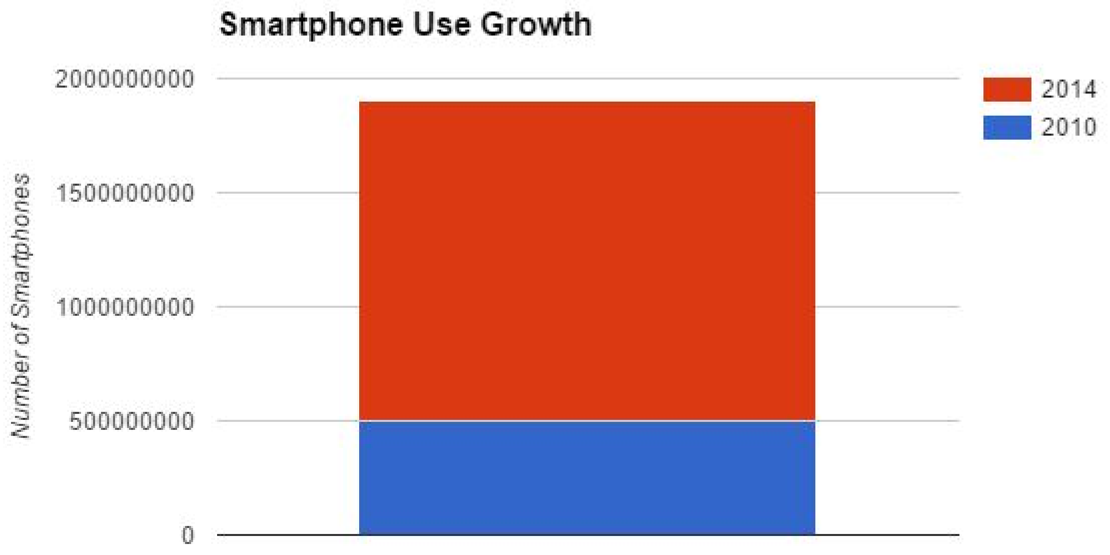
dietrichlehr@gmail.com

East Carolina University

## ABSTRACT

I will be focusing on mobile device security as it pertains to methodologies used to secure critical components of smartphone use. I will analyze and compare the two smartphone platform leaders, Apple and Android and the way they secure their devices. I will also provide insight as to which methods may be stronger and better for users to implement chose if multiple options are available.

The mobile device market has exploded in recent years. Smartphone use has increased by over 394 percent and tablet used by 1721 percent between 2010 and 2014. This rise marks a time where mobile usage has passed desktop use as the prefered method for online content. With mobile devices becoming the primary method for digital content access, it will become increasingly critical that the same security emphasis applied to desktops be applied to our handheld products as well.



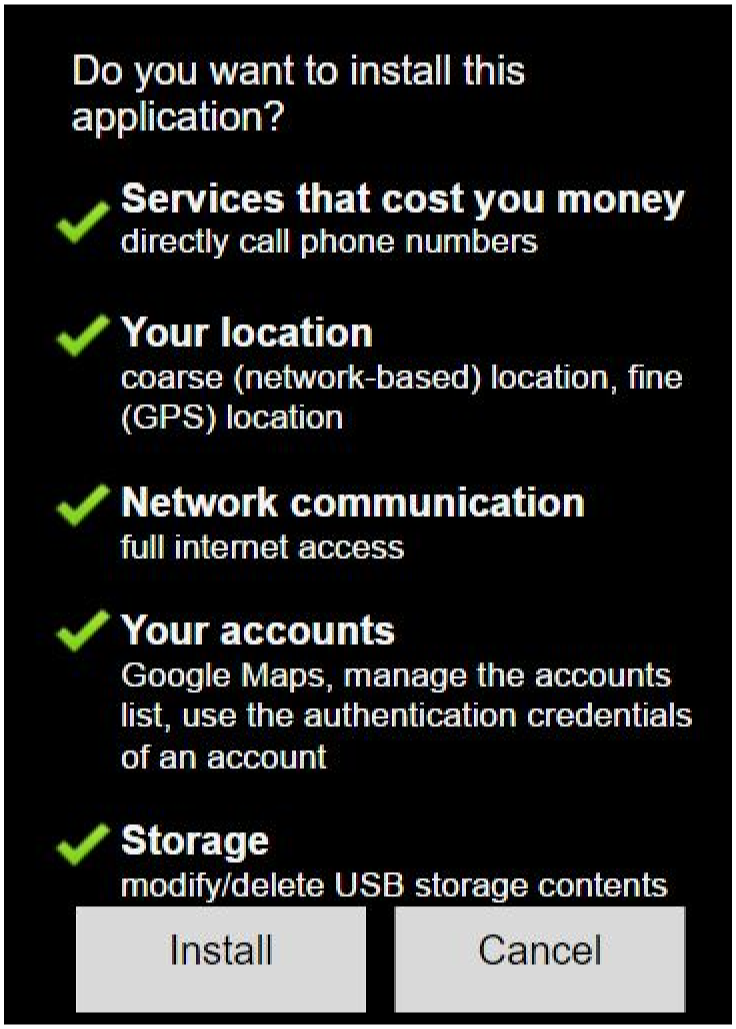
“Mobile security breaches have affected 68 percent of businesses” in 2015 (2015 Security Predictions). With such a drastic increase from the 45 percent from 2013, we see a dire need to secure our mobile devices. Some of the most commonly used mobile platforms are Google’s Android with 82 percent of the market share and Apple’s iOS with 13 percent (IDC). Windows holds roughly 2 percent of the market share while BlackBerry owns less than 1% (IDC).

Security experts agree that the focus in mobile device security should be on the applications primarily, since these are the primary means of accessing data on networks. This becomes a much larger issue when you consider that the pool of application developers isn’t limited to professionals. Hobbyist and novice alike are developing apps and publishing them for others to download at such a fast pace that simply securing the device wouldn’t keep it updated quickly enough. Developer toolkits also reduce the learning curve for application developers and don’t necessarily enforce security best practice techniques. Antimalware tool should also be implemented, but not in the traditional sense of signature-based scans, but with a focus on anomaly scans due to encryption and code transformation (Faruki).

The need for application focused security development is further highlighted by the fact that estimations predict 75 percent of application would fail the most basic security test (Gartner). Android has felt the pains of lacking security with the Stagefright vulnerability which affected 95 percent of all devices for a total of over 950 million. This attack was carried out by sending a multimedia or google hangouts message to a specific user, and is automatically ran without any input from the target (Critical Stagefright). The Open Web Application Security Project (OWASP) has released a top 10 mobile application risk list to identify areas in need of attention. One of the largest concerns is 3rd party applications “listening”, whether it’s to your voice by activating your microphone or recording data sent over text and email, the end result is a loss in data (OWASP). Two known applications that do this are the SMS Replicator on Android, and the SpyPhone on iPhone (OWASP). Another dangerous concern that is no different from the desktop realm is the rootkit such as the DroidDream trojan for Android (OWASP). These are exceptionally dangerous because they can modify the system to hide their presence and work undetected. Some experts suggest implementing a “zero trust” policy in which everything is denied by default and explicitly let in, similar to how Windows Firewall prompts you to allow new connections for the first time.

Apple has a multitiered approach to application security outlined in their iOS Security Guide. Apple enforces the use of signed certificates to be verified at run-time with certificates issued by them, similar to the way in which TLS/SSL works with verifying websites (Apple). This reduces the ability of an attacker to produce a trojan application and just place it in the app store since Apple has to approve and issue a certificate first. Before this developers can even apply for an Apple issued certificate, they must register with Apple through the iOS Developer Program (Apple). Just as if you were applying for a digital certificate, Apple goes through the process of verifying the applicant’s real-world identity. This acts as a deterrent to those who would prefer to remain anonymous when developing malicious apps. The major differentiation between Android and Apple is that Apple does not allow the installation of untrusted applications (Apple). This is an issue of security over usability for application developers who may want to develope for their own personal use, but Apple has deemed it not worth the risk. Apple implements the increasingly common sandbox technology to isolate apps and prevent them from attacking the system (Apple). “Application sandboxing enforces permissions, privileges, directories, entitlements and kernel access for a mobile app (Ahmad).” The iOS partition is read-only which prevents attackers from manipulating the OS even if they could obtain elevated permissions and all non-essential services have been omitted from iOS altogether (Apple). With a system this locked down, there would naturally be issues with applications being able to cross-talk. Apple addressed this by allowing what they call extensions, which are signed executable binaries that come with the app you download (Apple). The interesting point to make here is that the extensions are also sandboxed and kept completely separate from the apps that they were a part of further reducing the ability of one app to compromise another. In order to promote more security minded development, Apple has included APIs within their software development kit (SDK) that make implementing security features simple. This effort in standardization through APIs is critical to promote security among such a large group of developers. Apple also went a step beyond software verification and extended their certification program to any hardware accessory that could plug into the iPhone through their MFi program (Apple). This is done through a chip that Apple provides to authorized manufacturers that is embedded in the product and authenticates with the Apple device when plugged in using a certificate. This helps prevent the 3rd party hardware attack vector by minimizing what products can work with the device. This application and hardware implementation is an effective first layer of defense because all approval is done through Apple. The downside is that everything comes from one entity and results in more expensive hardware with strict development cycles and slower advances in technology.

Android has a fundamentally different idea behind it's OS platform. It focuses on the open source aspect utilizing a Linux kernel and the Java programming language rather than a proprietary one like Apple. Sandboxing, or isolating, applications is the primary security method for android devices (Security Tips). This provides a virtual instance in which the application can perform and limits the effectiveness of malware by restricting its access to system resources and other applications. User granted permission sets are also a form of security where users either allow or deny an application access to certain features on their device (Security Tips). Android secures sensitive functionality, such as sim card access, by purposely leaving out APIs to access it (Application Security). When APIs are necessary for access sensitive OS areas, permissions are used to grant access to items such as GPS, cameras, and bluetooth (Application Security). Apps present their required permissions when installed and the user has to chose to accept or deny them while keeping in mind denying can prevent the app from functioning as intended and the process is repeated even if the app is uninstalled and reinstalled.



Do you want to install this application?

✓ **Services that cost you money**
directly call phone numbers

✓ **Your location**
coarse (network-based) location, fine (GPS) location

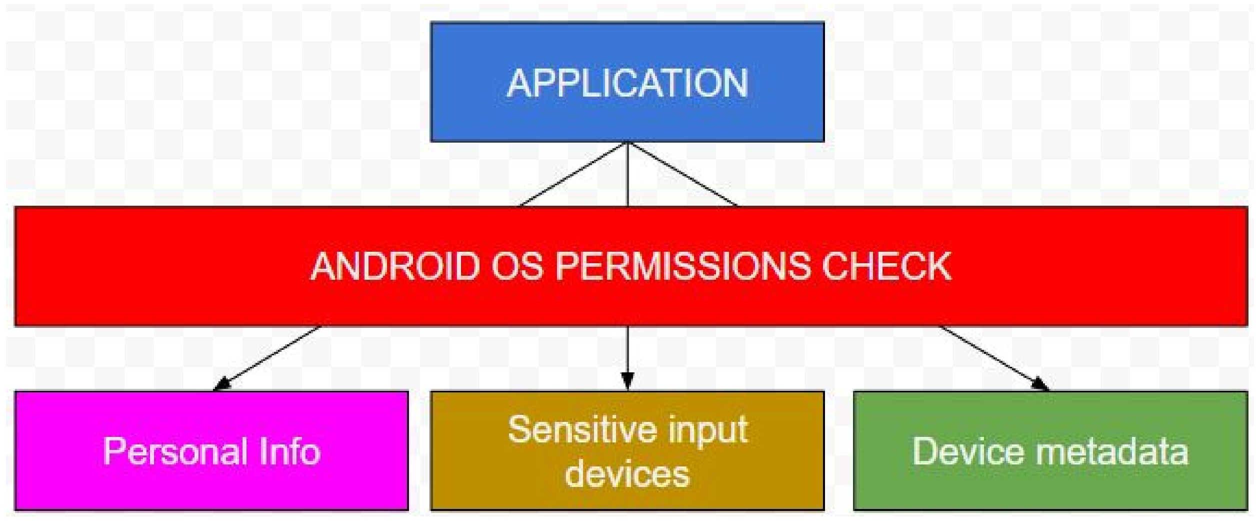
✓ **Network communication**
full internet access

✓ **Your accounts**
Google Maps, manage the accounts list, use the authentication credentials of an account

✓ **Storage**
modify/delete USB storage contents

Android's screen prompting user to allow application to access features

These permission notifications are crucial in an open source system where Android does not control every aspect of application development to inform users that the application they are using is requesting access to specific system resources. Android has also gone so far as to protect APIs that incur a "cost", meaning time increase, on a network segment (Application Security). This includes any messaging, phone call, or data use where latency may be an issue. Android also takes measures to secure the user’s personal data. All applications must pass through Android's permission checks before gaining access to personal data.



Permissions check flow in Android OS

Data is also restricted to the app that owns it, but it can be shared through Interprocess Communication (IPC) (Application Security). Google also requires app developers to digitally sign their applications which can be used to ensure that applications haven’t not been modified after being loaded to the Google Play store (Application Security). Certificate signing is enforced by Android’s package manager performs the verification process before an app can be installed. Some may view this as being less secure than the method Apple employs, but that is the tradeoff when you want to allow users greater control and flexibility using an open source model. Android allows the user to decide which apps they will install while providing information on what resources those apps will have access to so they may make an informed decision.

Device full-filesystem encryption is also highly encouraged. Android's implementation of full disk encryption is performed at the block level using the still unbroken AES encryption algorithm with either 128 or 256 bit. A user can choose to lock their android with either a PIN, password, or a pattern (Full Disk Encryption). Since full disk encryption is just that, the user is required to input their chosen number, word, or pattern at bootup. This is similar to the way a BIOS password works on a PC and is a very effective method in preventing unauthorized access. Even if a hacker was able to bypass a lock screen, he'd have to get past the full disk encryption before they had a chance. Below you can see how the process works.

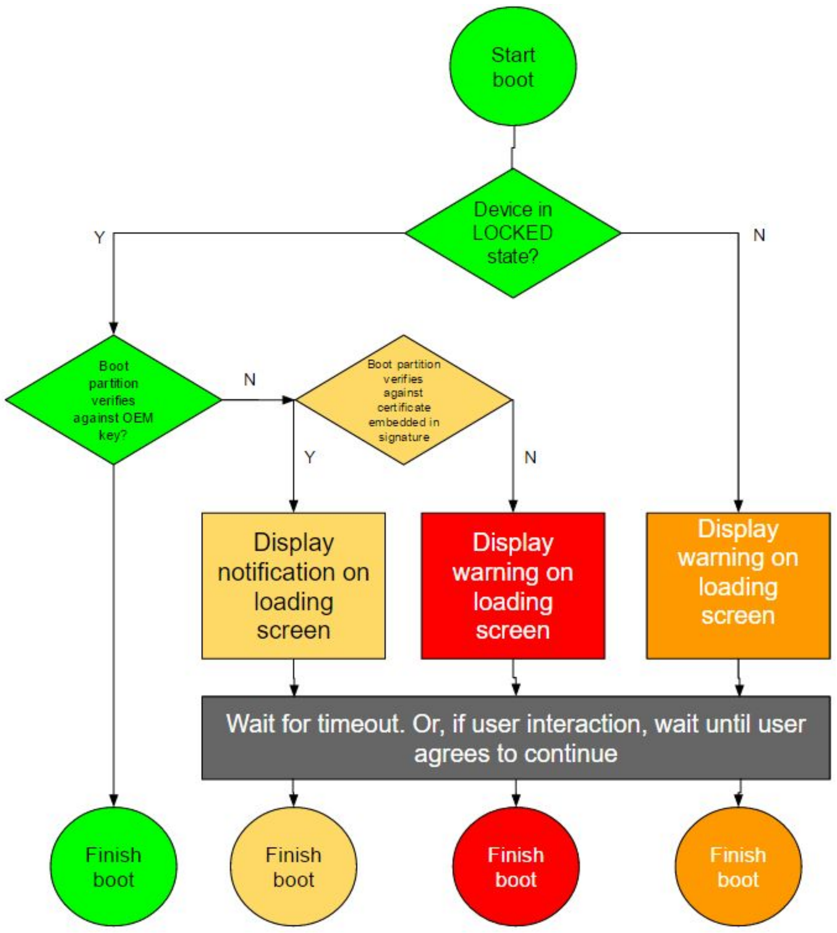

Start boot
Device in LOCKED state?
Y
N
Boot partition verifies against OEM key?
N
Boot partition verifies against certificate embedded in signature
Y
N
Display notification on loading screen
Display warning on loading screen
Display warning on loading screen
Wait for timeout. Or, if user interaction, wait until user agrees to continue
Finish boot
Finish boot
Finish boot
Finish boot

Image of Android boot sequence flow

Going back to a report in 2013, there were an estimated 3.1 million smartphones stolen with another estimated 1.4 million lost (Smart Phone Thefts). This is where full-filesystem encryption can help prevent data loss if you lose your device or might even deter a thief from stealing it in the first place.  Apple also employs similar technology utilizing the AES 256 bit encryption algorithm and complies with the Federal Information Processing Standards (FIPS) 140-2 Level 1 and is also seeking to meet the Common Criteria Certification (CCC) ISO 15408 standard (Apple).

For extra security, you should use some form of lock screen for regular access to your device while it’s on. This can be done with PINs, patterns, or even biometrics. The downside to the PIN is that they can be guessed and many users typically use the same numerical combinations, such as 1212 or 1234 according to studies (Unlocking Smartphones). Patterns also have their flaws. Researcher Marte Loge was able to find that 44 percent of patterns are started from the top left corner and move downward and to the right (Unlocking Smartphones). A pattern can also be easily observed by someone looking to gain access to your phone. Android has combated the shoulder-surfing technique by allowing you to disable the “make pattern-visible” option on the device.

The only option not susceptible to shoulder-surfing is the use of biometrics, which is now becoming standard on smartphone devices such as the iPhone 6 and the Nexus 5X and Nexus 6. The reason biometrics are so effective, is because they are something that you *are*. This makes guessing impossible as well as observing passcodes impossible. The only means an attacker would have to gain access would be to trick the user into compromising their device for them through the use of malware which could either bypass the fingerprint scan, or allow the theft of the biometrics database (Unlocking Smartphones).

Apple employs the use of biometrics through their system called Touch ID. Aside from the security factor, biometrics offer an element of convenience to accessing your phone. You can simple place your finger on a spot by feel and unlock it much quicker than you could if you had to look at it to enter your pin or pattern. This makes it more likely that a user will allow their phone to lock whenever they are finished with it rather than having extended timeouts which give attackers opportunities if they phone is left unattended. Apple summates that the likeliness of a false positive is 1 in 50,000, which is why they only allow 5 unsuccessful attempts to access a device with a fingerprint before you will have to provide a code (Apple).

Android's implementation of biometrics for device access is implemented through what they call the Fingerprint Hardware Abstraction Layer (HAL) (Fingerprint HAL). This is what allows the Android operating system to connect to a vendor's hardware fingerprint sensor. A user enrolls themselves with the Android's gatekeeper. After enrollment, a user can provide their fingerprint, which is then hashed and checked against the fingerprint that the user enrolled with to provide access to system resources

(Authentication). The user is then granted a token in a similar fashion to Microsoft's Kerberos ticket granting system.

Device security cannot be limited to smart phone access or applications alone. The information passing to and from the device must also be kept private to safeguard user's data. Since network security has to allow for the transmission of information to and from many different devices, standard protocols must be used. Apple and Android both use the same network security protocols with some slight variations for their own specific device communications. On iOS,TLS is the protocol used when applications must communicate over the network and the servers they contact are required to support certificates signed using either SHA-256 or better and must use an RSA key of either 2048-bit RSA or 256-bit elliptic curve key (Apple). The elliptic curve key is beneficial to the mobile market because it is far less resource intensive. If app's servers do not support these methods, the app will not be allowed to connect. Android also employs TLS to secure network communications between apps and servers.

Virtual Private Networks are also a sufficient means of securing network connectivity. Both Apple and Android support VPN use natively and they are very easy to configure with OpenVPN, PTPP, and L2TP/IPsec being the most commonly used methods of VPN implementation. A VPN works to encrypt all network traffic from between the device and its endpoint by creating a virtual tunnel for it to traverse. This hides the data even from the ISP whose network you are using, ultimately securing your data. Apple has further improved security by implementing a per app VPN, allowing each application to use its own VPN, further separating traffic (Apple). The use of VPNs is very popular among companies, because it allows a cheap but secure method for their employees to connect to the company's networked resources. Smartphones also support the Wi-Fi protocols, such as WPA2 with AES encryption. Some even support 802.1X, which allows authentication of the device's MAC address before it is even allowed to connect to the network. This requires pre registration with the company whose network you're connecting to so that they add your device information to their database. In addition to Wi-Fi support, Android and iOS both support Bluetooth to pair with devices such as cars, watches, speakers, etc. Bluetooth encrypts and authenticates using CBC-MAC with 128-bit AES block ciphers and devices utilizing version 4.0 or newer are considered Bluetooth low energy (RFC 7668). CBC-MAC encrypts the plaintext and then the encrypted result several times over to provide security.

Apple and Google have also implemented their own methods near field communication (NFC) called AirDrop and Android Beam respectively. AirDrop communication happens over Bluetooth and uses a 2048-bit generated RSA key (Apple). Devices transfer their hashes between each other to determine who they are and whether or not the should receive the payload (Apple). The default setting for AirDrop is to only allow communication with users who are in your contacts list which adds a layer of security if strangers are bluesnarfing in the area. Once authenticated, the devices create a Wi-Fi network and send each other's full hash to further identify the other end (Apple). If the user approves, a TLS session is created and the users can now send and receive their data (Apple). The key difference between Android Beam and AirDrop is that Android *only* uses Bluetooth, even for the data exchange. For this process to work, you pair it almost like any other Bluetooth device with the transfer being one way only and disconnecting from the paired device once complete (Advanced NFC). As with any service that you aren't using, AirDrop and Android Beam should always be disabled while you aren't using them in order to reduce attack vectors to your device. The same goes for Wi-Fi and Bluetooth features.

Today's smartphone devices are such an integral part of our everyday lives and have become sophisticated enough that we now use them as a form of payment. Apple's implementation of this is aptly called Apple Pay. The required pieces to make this work include a certified chip running the Java Card platform known as the Secure Element, the NFC controller, a wallet that holds your card and account information, and the Apple Pay servers which manage your wallet and encrypt your payment credentials (Apple). Once an iPhone user adds a form of payment to their wallet, it becomes available to pay with. It can be thought of as a credit card on your phone. With NFC enabled, a user can then place their phone next to an NFC terminal to initiate a payment. Through this NFC connection, the phone's secure element and the terminal communicate after the user provides credentials, whether fingerprint or PIN (Apple). This communication never extends beyond the phone's NFC field, preventing unauthorized access by applications (Apple). The use of AES encrypted nonces are used by the device and the terminal in order to prevent replay attacks and the transaction is processed (Apple). Google’s Android Pay functions in much the same way, but adds a level up abstraction to your account information by generating a 16 digit virtual number to exchange in its place. Unlike Apple, these tokens aren’t generated in a secure element on the phone itself, but are instead generated via cloud application with fallback capability to local tokens.

Apple and Google are increasing their efforts to secure their platforms during the mobile technology boom. Security is always an iterative process and with this industry having grown so rapidly, security is struggling to keep up, but new implementations like biometrics will help secure access to devices. Users should always restrict access to their phones with a PIN number at a minimum as patterns are considered less secure. Apple’s approach is to control the entire ecosystem that their phone operates in through stringent app control and even peripheral hardware, while Google allows the user to assume informed risk should they choose to install certain applications. It’s up to the user to decide which device best suits their needs and to adhere to recommended device security controls to safeguard their data when using them.

## References

2015 Security Predictions - Have They Held True So Far? | SecurityWeek.Com. (n.d.). Retrieved January 15, 2016, from http://www.securityweek.com/2015-security-predictions-have-they-held-true-so-fa

IDC: Smartphone OS Market Share. (n.d.). Retrieved January 15, 2016, from https://www.idc.com/prodserv/smartphone-os-market-share.jsp

Gartner Says More than 75 Percent of Mobile Applications will Fail Basic Security Tests Through 2015. (n.d.). Retrieved January 15, 2016, from https://www.gartner.com/newsroom/id/2846017

Gartner Says More than 75 Percent of Mobile Applications will Fail Basic Security Tests Through 2015. (n.d.). Retrieved January 15, 2016, from https://www.gartner.com/newsroom/id/2846017

Critical Stagefright Vulnerabilities Expose 950 Million Android Devices | SecurityWeek.Com. (n.d.). Retrieved January 15, 2016, from http://www.securityweek.com/critical-stagefright-vulnerabilities-expose-950-million-android-devices

OWASP. Retrieved January 15, 2016, from https://www.owasp.org/images/9/94/MobileTopTen.pdf

Apple. Retrieved January 15, 2016, from https://www.apple.com/business/docs/iOS_Security_Guide.pdf

Security Tips. (n.d.). Retrieved January 16, 2016, from https://developer.android.com/training/articles/security-tips.html

Application security. (n.d.). Retrieved January 16, 2016, from https://source.android.com/security/overview/app-security.html

Full Disk Encryption. (n.d.). Retrieved January 16, 2016, from https://source.android.com/security/encryption/index.html#how_android_encryption_works

Smart phone thefts rose to 3.1 million in 2013 - Consumer Reports. (n.d.). Retrieved January 16, 2016, from http://www.consumerreports.org/cro/news/2014/04/smart-phone-thefts-rose-to-3-1-million-last-year/index.htm

Unlocking Smartphones: PINs, Patterns or Fingerprints? (n.d.). Retrieved January 16, 2016, from http://www.esecurityplanet.com/mobile-security/unlocking-smartphones-pins-patterns-or-fingerprints.html

Fingerprint HAL. (n.d.). Retrieved January 16, 2016, from https://source.android.com/security/authentication/fingerprint-hal.html

Authentication. (n.d.). Retrieved January 16, 2016, from https://source.android.com/security/authentication/index.html

Advanced NFC. (n.d.). Retrieved January 16, 2016, from https://developer.android.com/guide/topics/connectivity/nfc/advanced-nfc.html

* RFC 7668 - IPv6 over BLUETOOTH(R) Low Energy. (n.d.). Retrieved January 16, 2016, from https://tools.ietf.org/html/rfc7668

* Faruki, P.; Bharmal, A.; Laxmi, V.; Ganmoor, V.; Gaur, M.S.; Conti, M.; Rajarajan, M., "Android Security: A Survey of Issues, Malware Penetration, and Defenses," in*Communications Surveys & Tutorials, IEEE* , vol.17, no.2, pp.998-1022, Secondquarter 2015
doi: 10.1109/COMST.2014.2386139
URL:
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6999911&isnumber=7110413

* Ahmad, M.S.; Musa, N.E.; Nadarajah, R.; Hassan, R.; Othman, N.E., "Comparison between android and iOS Operating System in terms of security," in *Information Technology in Asia (CITA), 2013 8th International Conference on* , vol., no., pp.1-4, 1-4 July 2013
doi: 10.1109/CITA.2013.6637558
URL:
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6637558&isnumber=6637544