EECS 710 - Fall 2012

**Mobile Application Security**

Himanshu Dwivedi

Chris Clark

David Thiel

Presented by

Bharath Padmanabhan

**The Authors**

**Himanshu Dwivedi** is a co-founder of iSEC Partners, an information security firm specializing in application security. He is also a

renowned industry author with six

security books

published.

**Chris Clark** is a principal security consultant at iSEC Partners, where he writes tools,

performs

penetra5on tests, and serves as a Windows and

Mobile expert.

**David Thiel** is a

Principal Security Consultant with iSEC Partners. His areas of exper5se are web application

penetra5on tes5ng, network protocols, fuzzing, Unix, and Mac OS X.

**Contents**

**Part I Mobile Platforms**

• Top Issues Facing Mobile Devices

• Tips for Secure Mobile Application Development

• Apple iPhone

• Google Android

**Part II Mobile Services**

• WAP and Mobile HTML Security

• Bluetooth Security

• SMS Security

• Mobile Geoloca8on

• Enterprise Security on the Mobile OS

**Part I Mobile Platforms**

**Top Issues Facing Mobile Devices**

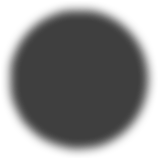
• Physical Security

• Secure Data Storage (on Disk)

• Strong Authentication with Poor Keyboards

• Multiple-User Support with Security

• Safe Browsing Environment • Secure Operating Systems  • Application Isolation  • Information Disclosure

• Virus, Worms, Trojans, Spyware, and Malware 

• Difficult Patching/Update Process

• Strict Use and Enforcement of SSL

• Phishing

• Cross-Site Request Forgery (CSRF)

• Loca8on Privacy/Security • Insecure Device Drivers • Mul8factor Authen8ca8on

**Top Issues Facing Mobile Devices**

**Physical Security**

• Loss of information from lost or stolen devices

• Unauthorized usage by the borrower

• Physical security has always meant little-to-no security

**Secure Data Storage (on Disk)**

• Sensi5ve information stored locally (password files, tokens, etc.)

• Prevent unauthorized access while making it accessible to certain applications on an as-needed basis

6

**Top Issues Facing Mobile Devices**

**Strong AuthenBcaBon with Poor Keywords**

• Password or passphrase that uses a combination of letters, numbers, special characters, and a space

• the Same standard on a mobile keyboard is difficult, if not impossible

**multiple-User Support with Security**

•, Unlike traditional client operating systems that support multiple users with different operating environments, no such thing as logging into a mobile device as a separate user

• No dis5nc5on between applications for business purposes vs. personal

• Need a unique security model by application to prevent data exposure

7

**Top Issues Facing Mobile Devices**

**Safe Browsing Environment**

• Lack of real estate makes phishing attempts easier

• Inability to view the entire URL or the URL at all

• Links are followed a lot more on mobile devices

**Secure operating Systems**

• Securing an OS is no easy task but should s5ll be undertaken by all mobile vendors

• Security open correlates to data loss but can also correlate to system down5me and diminished user experience

8

**Top Issues Facing Mobile Devices**

**Application Isolation**

• Very common to see various types of applications (corporate, gaming, social, etc) on a mobile device

• Ability to isolate these applications and the data they require is crucial

**information Disclosure**

• Data stored on a device (desktop, laptop, server, mobile) is worth more than the device itself, however, a mobile device more likely to be lost or stolen

• Access from a mobile device to other networks (say VPN) is another area of concern if authen5ca5on mechanisms are not strong

9

**Top Issues Facing Mobile Devices**

**Virus, Worms, Trojans, Spyware, and Malware**

• Mobile devices also face the threat of viruses, worms, Trojans, spyware, and malware

• Lessons to learn from the desktop world but also need to adjust to the mobile environment and new attack classes

**Difficult Patching/Update Process**

• Patching and upda5ng not a technical challenge but several considera5ons make it a difficult problem for mobile

• Carriers have big problems with immediate system updates and patching due to little response 5me for tes5ng

• Requires coordination among OS developers, carriers, and handset vendors

10

**Top Issues Facing Mobile Devices**

**String Use and Enforcement of SSL**

• Older devices lacked the horsepower to enforce SSL without affecting user experience; some s5ll allowed for backward-compa5bility

• Some organizations defaul5ng to clear-text protocols assuming increased complexity of sniffing on 3G network

• Abundance of transi5ve networks between the mobile devices and the end system

**Phishing**

• Users are more prone to clicking links on mobile without safety concerns

• Lack of real estate to show en5re URL or the URL itself

11

**Top Issues Facing Mobile Devices**

**Cross-Site Request Forgery (CSRF)**

• Big problem for mobile HTML sites that are vulnerable

• Easy to get vic5ms to click on links due to previously mentioned factors

• Allows attacker to update a vic5m's information (address, email, password, etc) on a vulnerable application

**LocaBon Privacy/Security**

• Most mobile users have assumed their local privacy was lost as soon as they started carrying a mobile device

• Users willingly give away their loca5on-specific information through applications like Google La5tude, Foursquare, etc.

12

**Top Issues Facing Mobile Devices**

**Insecure Device Drivers**

• Most applications should not have system access to mobile devices but device drivers need such access

• Exposure to attackers if third-party drivers provide methods to get around protec5on schemes via potentially insecure code

**MulBfactor AuthenBcaBon**

• SoT mul5factor authen5ca5on schemes (same browser, IP range, HTTP headers) used by mobile web applications are very vulnerable to spoofing

• Typical to create a device signature using a combination of HTTP headers and proper5es of the device's connection but still is not good enough compared to na5ve mobile applications

13

**Tips for Secure Mobile Application Development**

• Leverage TLS/SSL

• Follow Secure Programming Prac5ces

• Validate Input

• Leverage the Permissions Model Used by the OS

• Use the Least Privilege Model for System Access

• Store Sensi5ve Informa5on Properly

• Sign the Applica5on’s Code

• Figure out a Secure and Strong Update Process

• Understand the Mobile Browser’s Security Strengths and Limitations

• Zero Out the Non-Threats

• User Secure/Intui5ve Mobile URLs



**Tips for Secure Mobile Application Development**

**Leverage TLS/SSL**

• Turn on Transport Layer Security (TLS) or Secure Sockets Layer (SSL) by default

• Both confidentiality and integrity protections should be enabled; many environments often enforce confidentiality but do not correctly enforce integrity protec5on

**Follow Secure Programming Practices**

• Big rush (and a small budget) to get a product out the door, forcing developers to write code quickly and not make the necessary security checks and balances

• Leverage the abundance of security frameworks and coding guidelines available

15

**Tips for Secure Mobile Application Development**

**Validate Input**

• Validating input is imperative for both native mobile applications and mobile web applications

• Mobile devices do not have host-based firewalls, IDS, or antivirus software, so basic sanitization of input is a must

**Leverage the Permissions Model Used by the OS**

• Permissions model is fairly strong on the base device, however, an external SD card may not be as secure

• Applica5on isola5on provided by systems like iOS and Android should be leveraged

16

**Tips for Secure Mobile Application Development**

**Use the Least Privilege Model for System Access**

• The least privilege model involves only asking for what is needed by the application

• One should enumerate the least amount of services, permissions, files, and processes the application will need and limit the application to only those items

• The least privilege model ensures the application does not affect others and is run in the safest way possible

**Store SensiBve InformaBon Properly**

• Do not store sensi5ve information (usernames, passwords, etc.) in clear text on the device; use na5ve encryp5on schemes instead

17

**Tips for Secure Mobile Application Development**

**Sign the ApplicaBon’s Code**

• Although signing the code does not make the code more secure, it allows users to know that an application has followed the practices required by the device’s application store

• Unsigned applications may have a much-reduced number of privileges on the system and will be unable to be widely disturbed through the various application channels of the devices

• Depending on whether or not the application is signed, or what type of cer5ficate is used, the application will be given different privileges on the OS

18

**Tips for Secure Mobile Application Development**

**Figure Out a Secure and Strong Update Process**

• Much like in the desktop world, an application that is not fully patched is a big problem for the application, the underlying OS, and the user

• A secure update process needs to be figured out where an application can be updated quickly, easily, and without a lot of bandwidth

**Understand the Mobile Browser’s Security Strengths and limitations**

• Understand the limitations of cookies, caching pages locally to the page, the Remember Password check boxes, and cached creden5als

• Do not treat the mobile browser as you would treat a regular web browser on a desktop operating system

19

**Tips for Secure Mobile Application Development**

**Zero Out the Non-Threats**

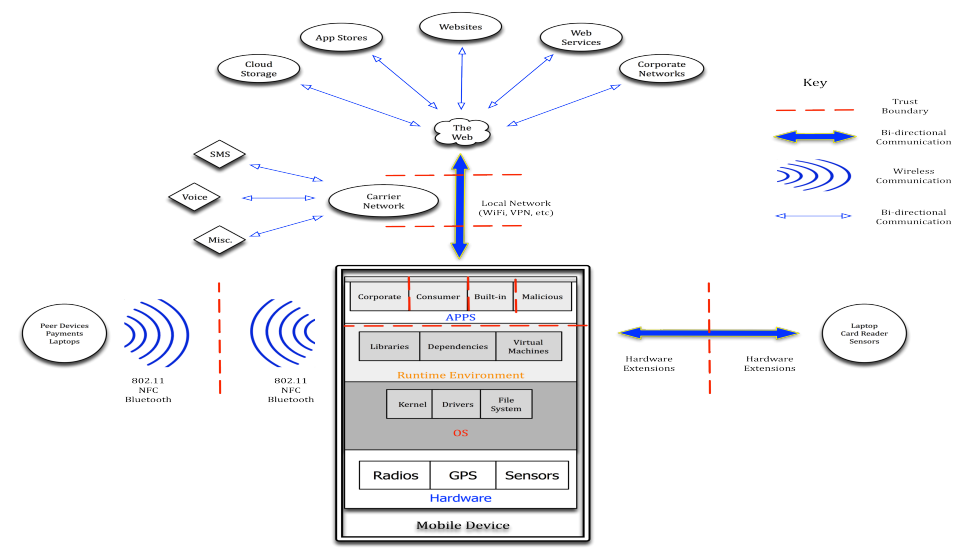
• Although the threats to mobile devices and their applications are very real, it is important to understand which one's maKer to a given application

• The best way to start this process is to enumerate the threats that are real, design mi5ga5on strategies around them, and note the others as accepted risks ("threat model")

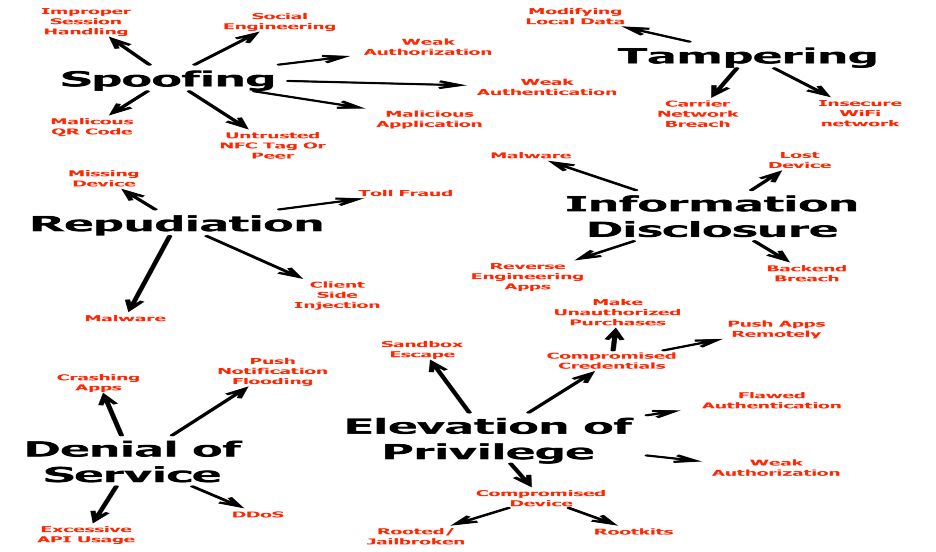
• Threat model should allow application developers to understand all the threats to the system and enable them to take action on those that are too risky to accept

20

**OWASP Mobile Threat Model**

****21

**OWASP Mobile Threat Model**

****22

**Tips for Secure Mobile Application Development**

**Use Secure/IntuiBve Mobile URLs**

• Some organizations use third-party to host their mobile sites whose domain will be different from the organization

• Many organizations have mobile-op5mized sites separate from their regular websites but it is important to keep the URLs intui5ve

**Intuitive**

• m.isecpartners.com

**Not So Intuitive**

• isecpartners.mobi

• isecpartners.mobilevendor.com

23

**Apple iPhone**

• Development 

• Security Tes5ng

• Applica5on Format

• Permissions and User Controls

• Local Data Storage

• Networking

• Push No5fica5ons, Copy/Paste, and Other IPC

**Development**

• Performed with Xcode and the iPhone SDK

• Can be run either within the emulator or on a physical device • Debugging is done within Xcode via gdb

• Objec5ve-C code can be decompiled fairly easily using standard OS X developer tools

• It is *not* possible to prevent reverse-engineering of the code 25

**Security Testing**

• Threat of classic C exploits is reduced, but not eliminated, by using high-level Objec5ve-C APIs

• To avoid buffer overflows, avoid manual memory management and use Cocoa objects such as NSString for string manipula5on (Integer overflows s5ll possible even when using NSInteger)

• Double-frees, where a segment of memory is already freed from use and an attempt is made to deallocate it again, is a problem

• Most commercial sta5c analysis tools haven’t matured to detect Objective-C-specific flaws, but simple free tools can be used to find C API abuses

26

**Application Format**

• Applications are compiled via Xcode using the GNU GCC compiler, cross-compiled for the ARM processor and local machine (emulator)

• Each application bundle includes a unique ID, a list of entitlements and preferences, a code signature, any required media assets, and the executable itself

• All iPhone applications have to be distributed through the “App Store” which has to be approved by Apple before distribution and can be revoked anytime at Apple’s discretion

• All iPhone applications have to be signed by a valid code-signing certificate; requires membership with the iPhone Developer Program

• On “jailbroken” iPhones, using Cydia and Installer are the two most popular ways to install unauthorized third-party software

27

**Permissions and User Controls**

• Apple uses “Mandatory Access Control” (MAC) as its mechanism for restric5ng the capabilities of applications

• The iPhone OS and OS X permission system (“sandboxing”) is based on the TrustedBSD framework which allows for writing policy files that describe what permissions an application should have

• Each application is installed into its directory (GUID); they are allowed limited read access to some system areas but not allowed to read/write directories belonging to other applications

• Both the heap and stack are non-executable by default; newer versions support ASLR (Address Space Layout Randomiza5on)

• Permissions gran5ng for specific func5onality (loca5on, contacts) is granted via popups to the user at the 5me of API use

28

**Local Data Storage**

• **SQLite Storage:** This is a popular way to persist application data but is subject to injec5on attack like any type of SQL database. Parameterized queries should be used to ensure third-party SQL is not accidentally executed by the application.

• **Keychain Storage:** The iPhone includes the Keychain mechanism from OS X (with some differences) to store creden5als and other data. The API is simpler compared to the Cocoa API and all the data is stored in a dic5onary of key/value pairs. It is to be noted that Keychain APIs only work on a physical device.

• **Shared Keychain Storage:** iOS 3.0 introduced this ability allowing for separate applications to share data by defining additional

“en5tlements”. The developer has to explicitly specify this when adding the attribute and also define the entitlement.

29

**Networking**

• **URL Loading API:** Supports HTTP, HTTPS, FTP, and file resource types using the NSURLConnec5on and NSURLDownload APIs with an NSURL object as the input. It is to be noted that HTTP and HTTPS request results are cached on the device by default and all cookies stored are accessible by any application that uses the URL loading system.

• **NSStreams:** Useful when using network sockets for protocols other than those handled by the URL loading system, or in places where you need more control over how connections behave.

• **Peer to Peer (P2P):** iOS 3.0 introduced this ability to do P2P networking between devices via Bluetooth. Opportuni5es for data they are increased since the game and non-game applications use it for collaboration and data exchange. Also, because data can potentially be streamed to the device by a malicious program or user, it is another untrusted input to be dealt with.

30

**Push Notifications, Copy/Paste, and Other IPC**

• **Push NoBficaBons:** iOS 3.0 introduced this ability allowing applications to provide users with no5fica5ons when they are not running. The device and push service platform perform mutual cer5ficate authen5ca5on. No5fica5on types can include popups or upda5ng the badge. It should be noted that push no5fica5ons are *not* guaranteed to be delivered.

• **UIPasteboard:** Similar to OS X, this can be implemented to handle the copying and pasting of objects within an application or to handle data to share among applications. Copied and pasted data is stored in item groupings with various representa5ons. Any information on shared pasteboards should be considered untrusted and potentially malicious and needs to be sani5zed before use.

31

**Google Android**

• Development 

• Platform Security Architecture

• Security Model

• Permissions

• Securable IPC Mechanisms

• Applica5on Signing

• Memory Management Security Enhancements • Files, Preferences, and Mass Storage

**Development**

• SDK provides free tools for building and debugging applications, supporting developers on Linux, Windows, and OS X

• SDK provides an emulator that emulates ARM-based devices and also alternate virtual hardware configura5ons

• Debugging support is built into Android and working with a device or with the emulator is mostly interchangeable

• Code developed using the SDK generally runs in the Dalvik VM 33

**Platform Security Architecture**

platforms by re-purposing tradi8onal opera8ng system security controls to:

• Android seeks to be the most secure and usable operating system for mobile

• Protect user data

• Protect system resources (including the network)

• Provide application isola8on

• To achieve these objectives, Android provides these key security features:

• Robust security at the OS level through the Linux kernel

• Mandatory application sandbox for all applications

• Secure interprocess (IPC) communication

• Applica8on signing

• Applica8on-defined and user-granted permissions

34

**Security Model**

• Android is based on the Linux security model with some abstrac5ons unique to it and leverages Linux user accounts to silo applications

• Android permissions are rights given to applica5ons to allow them to take pictures, use the GPS, make phone calls, and so on

• When installed, applications are given a unique user iden5fier (UID); the UID is used to protect an application’s data

• The need for permissions minimizes the impact of malicious software unless a user grants powerful rights to dubious software

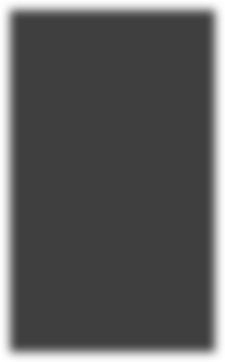
• Android’s run5me system tracks which permissions each application has; these permissions are granted either when the OS was installed or upon installa5on of the application by the user

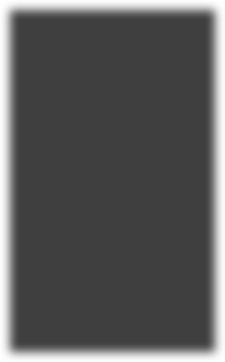
35

**Permissions**

• Android uses *manifest permissions* to track what the user allows applications to do, such as sending SMS, using the camera, etc.

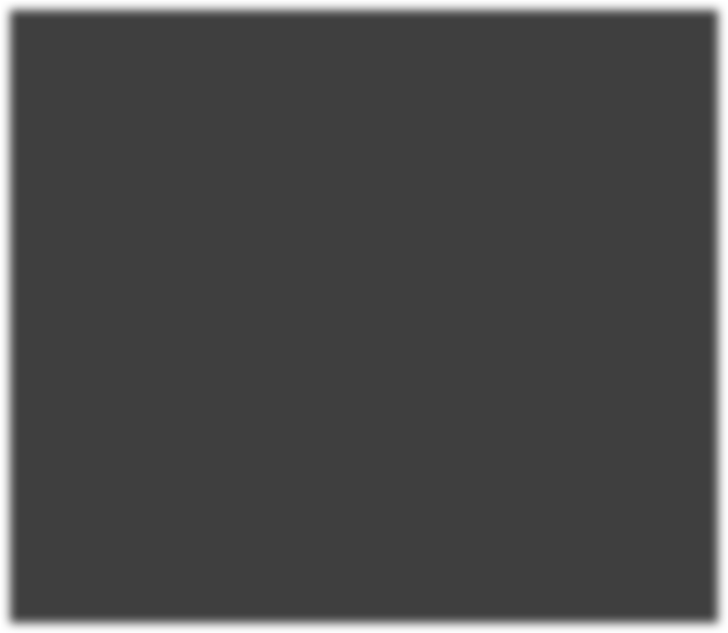
• Before installation of any application, the user has shown the different permissions the application is reques5ng. Once installed, an application’s permissions *cannot* be changed.

*Permissions at Applica1on Install* 

*Permissions of an Installed Applica1on*

36

**Permission Protection Levels**

****37

**Securable IPC Mechanisms**

• **AcBviBes** are interac5ve screens used to communicate with users. Intents are used to specify an Ac5vity.

• **Broadcasts** provide a way to send messages between applications. When sending a broadcast, an application puts the message to be sent into an Intent.

• **Services** are background processes that toil away quietly in the background.

• **ContentProviders** provide a way to efficiently share rela5onal data between processes securely. They are based on SQL.

• **Binder** provides a highly efficient communication mechanism. It is commonly used to bridge Java and na5ve code running in separate processes.

38

**Application Signing**

• Every application that is run on the Android platform must be signed by the developer. Applica5ons that attempt to install without being signed will be rejected by either Google Play or the package installer on the Android device.

• The signed application cer5ficate defines which user id is associated with which application. Applica5on signing ensures that one application cannot access any other application except through a well-defined IPC.

• Applica5ons can be signed by a third party or self-signed. Android provides code signing using self-signed cer5ficates that developers can generate without external assistance or permission.

39

**Memory Management Security Enhancements**

• ProPolice to prevent stack buffer overruns

• safe\_iop to reduce integer overflows

• Extensions to OpenBSD dlmalloc to prevent double free()  vulnerabili8es and to prevent chunk consolida8on a\acks

• OpenBSD calloc to prevent integer overflows during memory alloca8on

• Format string vulnerability protec8ons

• Hardware-based No eXecute (NX) to prevent code execu8on on the  stack and heap

• Linux mmap\_min\_addr to mi8gate null pointer dereference privilege  escala8on

• Address Space Layout Randomiza8on (ASLR) to randomize key locations in memory

40

**Files, Preferences, and Mass Storage**

• UNIX-style file permissions are present in Android; each application runs as its user so files created by one application cannot be read or altered by another application (unless the user allows it)

• SharedPreferences is a system feature that is backed by a file with permissions like any others

• Android devices may support larger add-on file systems mounted on memory cards since devices typically have a limited amount of memory

• Data stored on memory cards is unprotected and cannot be accessed by any program on the device

41

**Contents**

**Part I Mobile Pla:orms**

• Top Issues Facing Mobile Devices

• Tips for Secure Mobile Applica8on Development

• Apple iPhone

• Google Android

**Part II Mobile Services**

• WAP and Mobile HTML Security

• Bluetooth Security

• SMS Security

• Mobile Geoloca8on

• Enterprise Security on the Mobile OS

**Part II Mobile Services**

**WAP and Mobile HTML Security**

• WAP and Mobile HTML Basics

• Authen5ca5on on WAP/Mobile HTML Sites

• Encryp5on

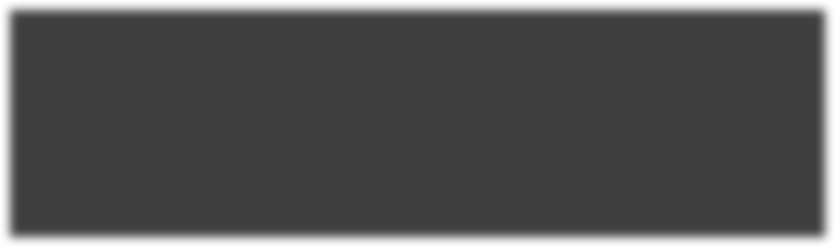
• Applica5on AKacks on Mobile HTML Sites

• WAP and Mobile Browser Weaknesses



**WAP and Mobile HTML Basics**

• WAP is a method to access the Internet from mobile devices • WAP gateway acts like a proxy server transla5ng content • WAP 2.0 does not require a WAP gateway



*WAP Architecture*

45

**Authentication on WAP/Mobile HTML Sites**

• One of the many problems that WAP and Mobile HTML developers have with mobile devices is the keyboard.



*PDA-style keyboard*

*Non-PDA-style keyboard*

• Strong passwords (consisting of letters, numbers, and special characters) are difficult to type on Non-PDA-style keyboards.

46

**Authentication on WAP/Mobile HTML Sites**

• Mobile PIN, an alternative to complex passwords, increases the user experience at the cost of lowering the security of the authen5ca5on process

• PIN typically consists of only 4-8 digit numbers, making it easier for brute-force attempts

• Crossover use of SMS and WAP/Mobile HTML applications is another avenue of exposure. Ex: Sending an SMS message to a predefined number will return an account balance as long as the caller ID value is correct.

• Spoofing of caller ID (“trusted value”) is quite simple

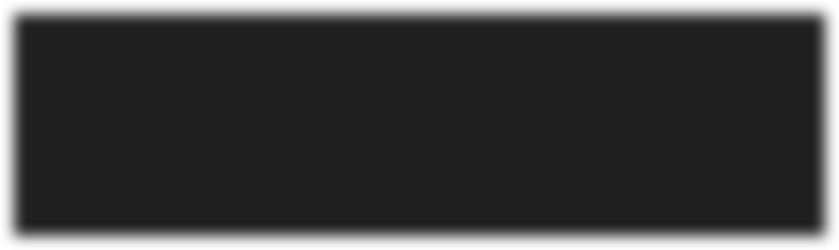
47

**Encryption**

• SSL/TLS is a critical aspect of online security to keep sensi5ve information private over the Internet

• WAP 1.0 used TLS but not end-to-end due to the limited horsepower on mobile devices

• WTLS is similar to TLS; used for low-bandwidth data channels that cannot support full-blown TLS implementation



*WAP 1.0 and transport encryp1on*

48

**Encryption**

• WAP 2.0 supports full end-to-end TLS to eliminate the “WAP gap” • WAP gateway op5onal (can be used for op5miza5on purposes) • WTLS is no longer needed



*WAP 2.0 and transport encryp1on*

49

**Application Attacks on Mobile HTML Sites**

• Many traditional web applications attacks will work on mobile browsers/devices supporting WAP 2.x/Mobile HTML sites

• Cross-Site Scrip5ng (XSS)

• SQL Injec5on

• Cross-Site Request Forgery (CSRF)

• HTTP Redirects

• Phishing

• Session Fixa5on

• Non-SSL Login

50

**WAP and Mobile Browser Weaknesses**

• Known limitations of WAP and mobile browsers:

• Lack of HTTPOnly Flag Support

• Lack of SECURE Flag Support

• Handling Browser Cache

• WAP Limita5ons

51

**Bluetooth Security**

• Overview of the Technology

• Bluetooth Security Features

• Pairing

• Authen5ca5on

• Authoriza5on

• Confiden5ality

• Threats to Bluetooth Devices and Networks • Bluetooth Vulnerabili5es

• Recommenda5ons

**Overview of the Technology**

• Conceived at Ericsson Mobile Communications to create a wireless keyboard system and then adapted for more generic purposes

• Common Uses include:

• Wireless keyboard, mouse, and printer connectivity

• Device synchronization (phone to desktop)

• File transfer (phone to desktop or photo printer)

• Gaming console integration (Nintendo Wii remotes and Sony PS3 headsets)

• Tethering for Internet access (using a data-enabled mobile phone as a modem for Internet access from a laptop with Bluetooth providing inter-device connectivity)

• Hands-free and voice-activated mobile phone kits for cars

53

**Bluetooth Security Features - Pairing**

• Pairing, the process whereby two Bluetooth devices establish a link and agree to communicate, is crucial to the overall security architecture and is tightly integrated with other security features

• During pairing, the communicating devices agree on and generate keys used to identify and relate to other devices; these keys are also used for device authen5ca5on and communication encryption

• Before Bluetooth v2.1, pairing between devices is accomplished through the entry of a PIN with a maximum length of 128 bits.

• Bluetooth v2.1 introduced Secure Simple Pairing to improve security through the use of Ellip5c Curve Diffie-Hellman for key exchange and link key generation

54

**Bluetooth Security Features - Authentication**

• Authen5ca5on is the process whereby one device verifies the identity of another device

• A traditional challenge-response mechanism is used between the *claimant* device and the *verifier* device

• Response to challenge based on a function involving a random number, the claimant’s Bluetooth device address, and a secret key generated during device pairing

• To prevent repeated attacks in a limited 5meframe, on an authen5ca5on failure the verifier will delay its next attempt to authen5cate the claimant

55

**Bluetooth Security Features - Authorization**

• Authoriza5on allows for decision-making about resource access and connec5on configura5on based on permissions granted a given device or service

• **Device Trust Levels:** Bluetooth devices can be “trusted” (previously been paired and have full access) or “untrusted” (not previously paired and have restricted access) with other Bluetooth devices

• **Service Security Levels:**

• Level 1 services require authen5ca5on and authoriza5on

• Level 2 services require authen5ca5on only

• Level 3 services have no security and are open to all devices 56

**Bluetooth Security Features - ConFidentiality**

• Confiden5ality is provided through the use of encryp5on

• Bluetooth uses E0, a stream cipher, as the basis for encryp5on and provides three different encryp5on modes

• Mode 1 does not do any encryp5on; all traffic is unencrypted

• Mode 2 encrypts traffic between individual endpoints but broadcast traffic is unencrypted

• Mode 3 encrypts both broadcast and point-to-point traffic 57

**Threats to Bluetooth Devices and Networks**

• Bluetooth devices and networks are also subject to threats like eavesdropping, impersona5on, denial of service, and man-in-the-middle attacks

• Addi5onal Bluetooth threats include:

• Loca5on tracking

• Key management issues

• Bluejacking

• Implementa5on issues (Bluesnarfing, Bluebugging, Car

whispering)

58

**Bluetooth Vulnerabilities**

• Bluetooth Versions Before v1.2

• The unit key is reusable and becomes public when used

• Bluetooth Versions Before v2.1

• Short PINs are permitted

• The encryp5on keystream repeats

• All Versions

• Unknown RNG strength for challenge-response

• Nego5able encryp5on key length (as small as one byte!)

• Shared master key

• Weak E0 stream cipher (theore5cal known-plaintext attack) 59

**Recommendations**

• Use complex PINs for Bluetooth devices

• In sensi5ve and high-security environments, configure Bluetooth devices to limit the power used by the Bluetooth radio

• Limit the services and profiles available on Bluetooth devices to only those required

• Configure Bluetooth devices as non-discoverable except during pairing

• Enable mutual authen5ca5on for all Bluetooth communica5ons • Configure the maximum allowable size for encryp5on keys

• Unpair devices that had previously paired with a device if a Bluetooth device is lost or stolen

60

**SMS Security**

• Overview of Short Message Service

• Overview of Mul5media Messaging Service

• Protocol AKacks

• Applica5on AKacks



**Overview of Short Message Service**

• SMS is designed for one mobile subscriber to send a short message (up to 160 characters) to another mobile subscriber



*SMS messages between phones using the*

*same carrier*

*SMS messages between phones on*

*different carriers*

62

**Overview of Short Message Service**

• A raw SMS message is known as a *Protocol Data Unit* (PDU)

• A basic SMS PDU contains several header fields as well as message contents

63

**Overview of Multimedia Messaging Service**

• MMS can send various types of images, audio, and video in addition to text

• MMS is fundamentally different from SMS although they may look the same from a user’s perspective



*MMS from a user standpoint Detailed MMS diagram*

64

**Protocol Attacks**

• Abusing Legi5mate Func5onality

• AKacks targe5ng func5onality that is meant to be hidden from the end user. Ex: Administra5ve and provisioning

communica5ons such as updates and voicemail no5fica5ons.

• AKacking Protocol Implementa5ons

• AKacks targe5ng vulnerabili5es in the implementation of the popular SMS protocols with the intent of sending a corrupted message to a vic5m’s phone resul5ng in the phone running hos5le code.

65

**Protocol Attacks - Abusing Legitimate Functionality**

• WAP Push AKack

• MMS No5fica5on

• BaKery-Draining AKack

• Silent Billing AKack

• OTA Sesngs AKack

66

**Application Attacks**

• Targets applications that use SMS as a delivery mechanism

•, Unlike protocol attacks that are mostly version agnos5c, application attacks are very specific to software versions running on phones

• Applica5on vulnerabili5es tend to fall into the browser, MMS client, or image categories

• Examples:

• iPhone Safari vulnerability results in heap overflow viewing a malicious page within mobile Safari, allowing an attacker to

execute arbitrary code on the iPhone

• Motorola RAZR JPG overflow vulnerability due to the way the RAZR parsed thumbprints in the JPG EXIF header, allowing

arbitrary code execu5on using malicious JPG image

67

**Mobile Geolocation**

• Geoloca5on Methods

• Geoloca5on Implementa5on

• Android

• iPhone

• Risks of Geoloca5on Services

• End User

• Service Providers

• Geoloca5on Best Prac5ces



**Geolocation Methods**

• Tower Triangula5on (Accuracy: 50m - 1,000m)

• Oldest widely used method of geoloca5on via cell phone

• Uses rela5ve power levels of radio signals between cell phone and cell tower; requires at least two cell towers

• Fairly inexact due to distance from cell towers and signal strength • GPS (Accuracy: 5m - 15m)

• Uses satellite signals instead of cell phone or wireless

infrastructure; recep5on may be poor indoors

• Can provide con5nuous tracking updates, useful for real-5me applica5ons

69

**Geolocation Methods**

• 802.11 (Accuracy: 10m - 200m but poten5ally erroneous)

• iPhone was the first smartphone to use this approach, using an API from Skyhook Wireless which uses data from wireless access points to create a large “wardriving” database

• Allows for devices without GPS to get potentially highly accurate local data

• Faster and more accurate than tower triangula5on

• Drawbacks due to dependency on wireless access points which could be moved

• Google’s “La5tude” service provides a newer implementation of Skyhook’s technology

70

**Geolocation Implementation - Android**

• Permission to use geoloca5on features is requested via the program manifest and is granted by the user at install 5me

• ACCESS\_COARSE\_LOCATION (for cell triangula5on or Wi-Fi) • ACCESS\_FINE\_LOCATION (for GPS)

*A permissions request for* 

*A permissions request for only fine local services*

*coarse and fine loca1on services*

71

**Geolocation Implementation - iPhone**

• Requires user approval every 5me an application that uses geoloca5on APIs is launched

• CLLoca5onAccuracyBest 

• CLLoca5onAccuracyNearestTenMeters

• CLLoca5onAccuracyHundredMeters

• CLLoca5onAccuracyThreeKilometers

*The iPhone loca1on*

*permissions dialog*

72

**Risks of Geolocation Services - End User**

• Posi5onal data stored on remote servers, when it can be 5ed to an individual, introduce a new avenue for data theT

• Along with other sensi5ve data, not only could this be a breach of user privacy, but also a potential source of information in court

• Broadcas5ng user’s loca5on voluntarily (think Foursquare) may also lead to stalking or harassment

• Few points to ponder:

• Privacy and data reten5on policies for posi5onal information • Third-party sharing and data transfer channels

• Course of action for law enforcement requests

73

**Risks of Geolocation Services - Service Providers**

• Risk of negative publicity from a data breach, legal or congressional subpoenas, and potential assistance to criminal acts by allowing third parties to track individual users

• OTen 5mes, the stored geoloca5on data is not necessary to provide the required func5onality

• Legal obliga5on to follow privacy guidelines in countries like the UK (“Data Protec5on Act”)

74

**Geolocation Best Practices**

• Use the least precise measurement necessary

• Discard data aTer use

• Keep data anonymous

• Indicate when tracking is enabled

• Use an opt-in model

• Have a privacy policy

• Do not share geoloca5on data with other users or services • Familiarize yourself with local laws

75

**Enterprise Security on the Mobile OS**

• Device Security Op5ons

• PIN

• Remote Wipe

• Secure Local Storage

• Encryp5on

• Applica5on Sandboxing

• Applica5on Signing

• Buffer Overflow Protec5on



**Device Security Options - PIN**

• Enabling the PIN is the first step in securing a mobile device

• Unmo5vated attacker could wipe and sell it instead of trying to break into the OS

• Data on the device (or data that the phone has access to) is at 5mes worth more than the device

• Although a four-digit PIN only needs 10,000 attempts to brute-force it, many mobile devices have a 5me delay after ten failed attempts

• On some devices like the iPhone, the SIM card also has PIN protec5on

77

**Device Security Options - Remote Wipe**

• The ability to remote wipe data on a mobile device (especially if its a corporate one) is impera5ve

• Remote wipe func5onality makes the loss of such devices a lot less stressful

• Both iPhone and Android support remote wipe func5onality 78

**Secure Local Storage**

• Ability to store sensi5ve information locally in a secure fashion is also an impera5ve security feature for mobile devices

• Many applications store login information, such as username and password, locally on the device in clear text (without encryp5on)

• The iPhone addresses this need via the use of a “Keychain” which can be used to store, retrieve, and read sensi5ve information, such as passwords, cer5ficates, and secrets

79

**Encryption**

• Full Disk Encryp5on

• Unlike desktop OSes, mobile OSes have little or no solutions for full disk encryp5on.

*• iOS4 and Android ICS supposedly have this feature*

• Email Encryp5on

• None of the most popular mobile OSes provide na5ve support for local email

*• Good for Enterprise supports both iPhone and Android*

• File Encryp5on

• Most major mobile OSes support file encryp5on

80