

COVER PAGE

Content Owner:

Owning Department: Corporate AQC
Local Business Unit: Corporate AQC
Business Department: Not Applicable

Description: The purpose of this standard is to optimize Mobile Applications by defining a consistent set of development requirements. Adherence to these requirements will ensure that applications are designed and built to minimize exposure to known security attacks.

Printed by: cdmsview (cdmsview) **for** cdmsview (cdmsview) **on** 25 Apr 2016 (GMT)

Effective: 18 Dec 2012 16:22:58 GMT

APPROVAL AND AUTHORISATION

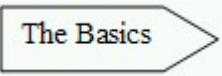
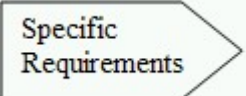
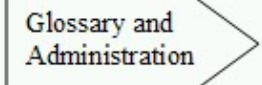
Completion of the following signature block has been carried out by Electronic Signature.

Document Approval:

Signed By : Burrell, Jon (ajb36243)
Decision : Approved
Decision Date : 18 Dec 2012 16:21:02 GMT
Role : Author
Purpose : Issue new Mobile Computing Standards
Meaning Of Signature : This document is suitable for issue.

Signed By : Burrell, Jon (ajb36243)
Decision : Approved
Decision Date : 18 Dec 2012 16:21:52 GMT
Role : Approver
Purpose : Issue new Mobile Computing Standards
Meaning Of Signature : This document is suitable for issue.

End of Cover Page

IT Standard		Title: Mobile Application Development Standard	
Official Short Title: MADS			
Key Points: <ul style="list-style-type: none"> ➤ Mobile applications must use controls to minimize the likelihood of exploit to known vulnerabilities in order to preserve the availability, confidentiality and integrity of IT systems and information. ➤ These standards apply to both in-house developed applications and commercial applications. ➤ Personnel must ensure that mobile applications delivered on behalf of GSK are security vulnerability free, either by use of this or commensurate industry standards. 			
Why do we have this standard? To ensure both consumer and enterprise mobile applications are developed securely to protect GSK customer, Confidential and Highly Confidential Information appropriately.			
What does this standard say?		Who in GSK has general obligations under this policy?	
	1. Purpose 2. Scope 3. Responsibilities	All GSK Staff <input type="checkbox"/>	
	4. Applicability 5. Controls	What functions in GSK have specific obligations under this policy?	
		Audit, Compliance, & Quality <input checked="" type="checkbox"/> Communications <input type="checkbox"/> Govt. & External Affairs <input type="checkbox"/> Finance <input type="checkbox"/> Global Procurement <input type="checkbox"/> HR <input type="checkbox"/> IT <input checked="" type="checkbox"/> Legal <input type="checkbox"/> Manufacturing & Supply <input type="checkbox"/> Marketing, Sales & Support <input type="checkbox"/> Medical <input type="checkbox"/> Supervisors & Management <input type="checkbox"/> Senior Management <input type="checkbox"/> Other <input type="checkbox"/>	
	6. Glossary 7. Administration 8. Waivers 9. Related Documents	Contacts: Any questions, issues or violations related to this standard should be communicated to IT Quality, Risk & Compliance via email address IT-QRC@gsk.com .	

The Basics

1. Purpose

The purpose of this standard is to optimize company-managed and software-as-a-service (SaaS) Mobile Applications by defining a consistent set of development requirements. Adherence to these requirements will ensure that applications are designed and built to minimize exposure to known security attacks in accordance with GSK IT Security Policy (POL-IT-0001) and overriding policies such as Company Assets and Proprietary Information (GSK-POL-100).

2. Scope

This standard applies to all mobile applications owned, maintained, created, leased or inherited by GlaxoSmithKline. Mobile applications developed by Outsourced IT Suppliers must ensure that the risks addressed in this standard are mitigated through compliance with the controls within, or through compliance with guidance from the Open Web Application Security Project at <http://www.owasp.org>. These may be applications intended for use by consumers, employees and business partners.

This standard applies to all platforms. Additional requirements and implementation details are provided for iPhone and iPad (“iOS”) and Android development platforms.

GSK mobile apps are likely to be externally exposed or accessible to third parties, either through commercial application stores, or through Internet-accessible GSK portals. It is also recognized that mobile devices are attractive targets for theft or compromise through malware, and are also more likely to be lost than other end user devices. Theft and lost devices also increase the exposure of GSK mobile applications to unauthorized access. For these reasons, this standard focuses on standards in alignment with GSK’s risk based approach to development.

This standard does not apply to mobile apps purchased by individual users from public sources (e.g. Apple or Google). Individual users must take into consideration GSK policies when considering use of any application that will access, process or store GSK proprietary information.

Reference the GSK End User Device Standard addresses requirements for controls that are necessary for mobile devices.

The Specifics

3. Responsibilities

IT Risk Management Compliance Board	Review this standard periodically for continued suitability for GSK and recommend any revisions. Define process to approve any exceptions to this standard.
Business Unit and Shared Service IT Architects	Accountable for ensuring that mobile applications developed within their organisations are compliant with this standard.
Business Unit IT personnel	Accountable for ensuring mobile applications developed by contracted third parties are compliant with this standard, including that any contracts with external agencies or developers for development of mobile applications include terms to ensure compliance with this standard or a commensurate industry standard defined by the Open Web Application Security Project.
Project Architects and Managers	Accountable for ensuring that this standard is appropriately complied with on projects through requirements and design.
Developers	Accountable for code development that complies with this standard.

4. Applicability

Individual controls in this document apply to mobile applications based on the type of application, the relative risk of the application, who provides the application and how much control GSK has over the inner workings of the application and which tier of a standard nTier architecture.

Controls are applied to mobile applications based on the impact of a failure or penetration into the applications. Controls are applicable based upon whether the application is categorised as a high risk application, medium risk application, or low risk application.

- High Risk Applications are applications that contain Highly Confidential Information, or are Internet Facing Systems that submit Personally Identifiable Information (PII) back to GSK.
- Medium Risk Applications are those that are GSK Critical Applications, or support Sarbanes-Oxley regulated business processes, contain Confidential Information, or are Internet Facing Systems (not containing PII).
- Low Risk Applications are those applications that are not high risk or medium risk.

While a control may not be applicable for a certain level of application risk or has a particular exemption, it should still be considered best practice for other applications even where it's not mandated.

Controls are also applied to mobile applications based upon the application's general design approach taken and supported features. The following three general categories have been defined that may influence a control's applicability for a given application type:

- **Native Mobile Applications** are platform-specific and developed using the mobile platform's Software Development Kit (SDK). Native applications may be more expensive and timely to develop, but may also deliver improved user experience and provide access to more native device features. Since Native applications can also implement custom web browser components, many traditional web application security risks can also apply to Native mobile applications.
- **Mobile Web/HTML5 Applications** are written entirely in HTML, CSS, and JavaScript. Mobile Web applications run in standard mobile web browsers and have the most in common with traditional web applications. Mobile Web applications provide a lightweight cross-platform solution to mobile application development, but may be less powerful than Native or Hybrid applications. Additionally, HTML5 applications do not require approval or distribution from the App Store or Android Market, as they are essentially mobile-enabled web applications. As a result, some security controls/risks in this document that apply to native platform components may not be applicable to pure Web/HTML5 based applications.
- **Hybrid Mobile Applications** use a combination of Web and Native components, allowing developers to use web technologies to access to native platform APIs and device features. Hybrid applications allow developers to leverage existing web development skills without having deep knowledge of native platform development. Since Hybrid components make use of both native and web components, many traditional web application security risks and native platform-specific risks generally apply to Hybrid mobile applications.

4.1. Applicability of Controls

The following table is provided as an aid to navigate the various control statements.



Tier	Control ID	Application Risk: / Control Name
Controls for iOS and Android	MAD S5.1.1	Mobile applications must comply with GSK Brand guidelines
	MADS5.1.2	Applications must display localized or unambiguous information
	MADS5.1.3	Mobile applications must be published through approved GSK services
	MADS5.1.4	Protect GSK Confidential application data on the device
	MADS5.1.5	Applications must not request more permissions than those which are necessary
	MADS5.1.6	Protect against buffer overflows, integer overflows, and format string attacks
	MADS5.1.7	Mobile applications that utilise cookies must implement them securely
	MADS5.1.8	Applications must follow secure memory management practices
	MADS5.1.9	Applications must not include hard-coded credentials and keys in application binaries and source code
	MADS5.1.10	Applications must secure copy and paste functions
	MADS5.1.11	Applications must build SQL queries use prepared statements and parameterized and validated
	MADS5.1.12	Applications must manage password credentials and authentication data securely
	MADS5.1.13	Applications must terminate sessions and remove authentication data in a secure manner
	MADS5.1.14	Applications must implement strong transport layer encryption and strict SSL/TLS validation
	MADS5.1.15	GeoL services must be used in a secure manner
	MADS5.1.16	Applications must perform strong input and output validation for all data received by the application and rendered to the user

Tier	Control ID	Application Risk: / Control Name
	MADS5.1.17	Applications must avoid detailed error messages that are useful to an attacker
	MADS5.1.18	Select an appropriate-strength authentication process in accordance with Appendix A.
	MADS5.1.19	Applications must enforce authorization on all protected resources
	MADS5.1.20	Mobile applications must display an approved End-User License Agreement (EULA) and obtain consent for the submission of PII to GSK.
Controls for iOS	MADS5.2.1	Disable caching of sensitive data within Apple's dynamic dictionary log
	MADS5.2.2	Disable automatic caching of iOS application screenshots
	MADS5.2.3	Custom URL schemes must be implemented in a secure fashion
	MADS5.2.4	iOS applications must not use the device's Unique Identifier (UDID) for authentication purposes
	MADS5.2.5	Mobile applications that deliver content in embedded web pages must protect against UI spoofing attacks.
Controls for Android	MADS5.3.1	Android applications must protect their components from unauthorized access and abuse
	MADS5.3.2	Android applications must not make their components "exportable" to other applications
	MADS5.3.3	Android applications must not be released with the "debuggable" flag enabled
	MADS5.3.4	Android applications must use code obfuscation techniques to protect source code from disclosure
	MADS5.3.5	Files opened or created on Android device must use secure permissions

5. Controls



5.1. Controls for iOS and Android mobile applications

5.1.1. Mobile applications must comply with GSK Brand guidelines



Description:	Where the GSK Name, Brand & Logotype and other branding components are used in mobile applications, they must be used in accordance with: <ol style="list-style-type: none"> GSK's On-line Brand Management Resource: "<i>Brand Focus</i>" (see below)
Exemptions:	Application components not displaying the GSK brand components described in GSK's On-line Brand Management Resource: " <i>Brand Focus</i> ".
Applies To:	Platforms:   Application Type: Hybrid: <input checked="" type="checkbox"/> , Native: <input checked="" type="checkbox"/> , Web/HTML5: <input checked="" type="checkbox"/> Application Risk: Low <input checked="" type="checkbox"/> , Medium <input checked="" type="checkbox"/> , High <input checked="" type="checkbox"/>
Rationale:	Our corporate identity guidelines form the foundation of the GSK brand communication strategy. The brand management components use various design elements within a set framework to achieve visual consistency. They are designed to allow creativity while upholding our brand values to achieve a consistent corporate identity wherever GSK branding is displayed.
See Also:	GSK's Brand Resource Web Site: http://brandresource.gsk.com/af.html
Examples:	Example of applications which might not use the GSK brand components: <ul style="list-style-type: none"> Internal business applications used solely by GSK staff Commercial off-the-shelf packages without pages which are intended to be configurable by or on behalf of the customer. Examples of GSK brand components include but are not limited to: <ul style="list-style-type: none"> Reproduction of the GSK logo in various sizes and formats Positioning and space requirements around GSK logos GSK Brand colour palette

	<ul style="list-style-type: none"> • GSK Brand typography and typefaces • GSK business naming
--	---

5.1.2. Applications must display localized or unambiguous information



Description:	<p>1) When displaying Date or Time, applications must either:</p> <p>a) Display Date and Time data in an unambiguous format.</p> <p>OR</p> <p>b) Detect the user's locale and display Date and Time according to the user's locale formatting convention,</p> <p>2) When displaying Monetary values, the currency must be unambiguous.</p>
Exemptions:	<p>Applications used solely within a single country or cultural group may use local conventions.</p> <p>COTS applications.</p>
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>The centralization of global applications will mean sharing of applications by an extended GSK workforce across geographies and cultures.</p> <p>As a global organisation GSK operates in many countries and uses many local currencies, therefore all monetary values must clearly indicate the denomination.</p>
See Also:	
Examples:	<p>The GSK preferred format for displaying date is DD-MMM-YYYY.</p> <p>The \$ symbol is used by several countries and therefore should be carefully differentiated by including the country reference. E.g. USD, AUD, CAD, for United State, Australian, and Canadian dollars respectively.</p>

5.1.3. Mobile applications must be published through approved GSK services

Description:	<p>All mobile applications intended for public consumption (e.g. to be used by general public consumers) must be published through the Apple App Store, Android Market and Nokia Ovi Store by the GSK Application Stores services. This service can be accessed by emailing a request to digital@gsk.com</p> <p>Mobile applications that are intended for in-house /non-public access must be published through GSK's corporate AppCatalog managed by EIS. The GSK corporate AppCatalog is for in-house non-public / non-consumer apps only</p>
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Publishing through the GSK Corporate Accounts ensures transparency to our customers that GSK is the developer of the app. This also enables GSK rather than engaged third parties to decommission or change mobile applications at any time and maintain the source code.</p>
See Also:	
Examples:	

5.1.4. Protect GSK Confidential application data on the device



Description:	<p>In order to protect against unauthorized access and disclosure, the following principles must be followed for all mobile applications that process Confidential or Highly Confidential GSK data:</p> <ol style="list-style-type: none"> 1. GSK Highly Confidential data must be stored on the server-side and never retained on the mobile device. 2. GSK Confidential data must be stored in an encrypted form that complies with GSK Global IT Cryptographic Standard.
---------------------	---

	<p>3. The following data must never be logged and retained in client-side log files. Sensitive data includes:</p> <ul style="list-style-type: none"> • Application debugging information, SQL queries, • Any functions and data related to the application's authentication/authorization mechanisms, • Function calls related to cryptographic functions, • Session tokens and encryption keys , and • Cached HTTP(S) requests and calls to backend web services.
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Application data that is insecurely stored client-side presents a significant exposure to consumer and Confidential GSK data. Failure to properly secure client-side data using strong encryption and hashing mechanisms can allow an attacker with physical possession of the device or mobile malware to compromise sensitive application data.</p> <p>Incorrectly coded mobile applications can leave sensitive data at rest unprotected in a variety of locations, including:</p> <ul style="list-style-type: none"> • Configuration and property files • Files created in the application's container and on external storage cards • Attachments saved within the application, data stored in local SQLite databases • client-side logs • Other cached data such as browser cookies and HTTP requests. <p>A common assumption made by developers is that private application data is sufficiently protected by the mobile platform's sandboxing model, in which private application files are stored in locations not generally accessible to casual users and third party applications. In the event of a device loss or theft, and the ability for an attacker to "root" or "jailbreak" the device, the sandboxing model and built-in OS protections are defeated, allowing retrieval of sensitive application data from the device.</p>

See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 – Insecure Data Storage - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project http://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/StandardBehaviors/StandardBehaviors.html http://developer.android.com/reference/javax/crypto/package-summary.html http://www.bouncycastle.org/ https://guardianproject.info/code/sqlcipher/
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>The Apple Data Protection API provides an additional layer of protection for files and Keychain items on the device by encrypting them with a combination of the user’s passcode and device-specific encryption keys. This feature allows developers to control when files and Keychain items can be decrypted and made accessible to applications and the underlying operating system.</p> <p>When writing sensitive files to non-volatile storage, the application should set the NSFileProtectionKey attribute to a value of NSFileProtectionCompleteUnlessOpen, in order the make files accessible only when the device is unlocked.</p> <p>When using the Keychain to store credentials on the device it is also possible to define accessibility settings for Keychain items, and control whether Keychain items can be migrated to another device. When adding sensitive credentials to the Keychain a protection class of kSecAttrAccessibleAlways, kSecAttrAccessibleAfterFirstUnlock, and kSecAttrAccessibleAlwaysThisDeviceOnly should never be used as these protection settings make Keychain secrets accessible even when the device is locked. Applications that make use of the Keychain to store credentials should use the most restrictive protection class available, kSecAttrAccessibleWhenUnlockedThisDeviceOnly, which makes Keychain items inaccessible when the device is locked and does not allow migration to other devices.</p> <p>Note: In the event of a lost device that is Jailbroken by an attacker, it is still possible to bypass the device’s passcode protection and gain access to all sensitive data on the device protected with Apple’s Data Protection API. As a result, applications that handle Confidential or Highly Confidential GSK data must not rely solely on the Data Protection API for sufficient data protection. While it does provide additional protection measures, more advanced security controls may be needed, such as jailbreak detection and remote wipe capabilities, to provide reasonable assurance that highly</p>

	<p>sensitive data is protected at rest.</p> <p><u>Android Guidelines:</u></p> <p>Several options are available for encryption and decryption of data at rest within Android applications, including the standard Java javax.crypto cryptographic packages. Another alternative is to use well vetted open source libraries such as the Bouncy Castle Crypto APIs, with extensive support for industry standard encryption algorithms and ciphers.</p> <p>Based on the use case, it may also be practical to use third party encryption tools such as SQLCipher for Android, which provides industry-standard encryption for local SQLite databases.</p>
--	---



5.1.5. Applications must not request more permissions than those which are necessary

Description:	<p>Applications must only request access to permissions and mobile OS features that are required to support the application's intended functionality. For example, applications that do not make calls or require access to the camera, GPS and external SD storage should not request such access. While most applications require access to some potentially sensitive features based on defined use cases, permissions must be carefully evaluated during the design phase.</p>
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Given the large number of native device features available to mobile applications, it is important to follow the principle of least privilege to protect the user's privacy and also protect against unauthorized access to data and misuse of device features available through the application's privileges.</p> <p>Confidential GSK data (including Personally Identifiable Information) could become compromised if the application runs with more privileges than necessary. Applications</p>

	that surreptitiously collect personal information, including but not limited to SMS logs, Contact Lists, and Geo-location data introduce serious privacy concerns and could expose GSK to reputational harm, especially under the EU Directive, if explicit consent is not received for its collection and use. Additionally, it may be possible for malicious third party applications to exploit security weaknesses in GSK applications and gain access to private data or functionality through an application IPC mechanism.
See Also:	
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>With Apple's permission model, third party applications run as the same "mobile" user and under the context of a restricted "container" sandbox profile. The sandbox profile controls access to system resources and operations at the kernel level, and does explicitly allow all applications access to some personal data on the device such as Address Book. iOS applications must request additional permissions via pop-ups to perform operations such as accessing the user's location</p> <p>As part of the App Store review process, Apple will typically challenge applications that call private APIs and other dangerous functions. However, iOS applications can still be approved to access private user data and device functionality, and care should be taken to remove unnecessary API requests from iOS applications.</p> <p><u>Android Guidelines:</u></p> <p>Within the Android platform each application runs as a separate Linux user, and granular permissions requested by the application are explicitly defined in the AndroidManifest.xml file and presented to the user upon installation. Using this model, the developer has granular control over application permissions in Android applications, and should never request privileges to access personal data or device features unless a strong business case exists.</p>

5.1.6. Protect against buffer overflows, integer overflows, and format string attacks

Description:	Mobile applications written in native code (C/C++/Objective-C) or containing packaged libraries written in native code must protect against security flaws such as buffer overflows, integer overflows, and format string attacks.
---------------------	--

	<p>GSK mobile applications written in native code must address the following:</p> <ul style="list-style-type: none"> • Use higher level interfaces available in the programming language to avoid overflows, including: <ul style="list-style-type: none"> ○ For code written in C, use Core Foundation CFString objects; ○ For code written in C++, use ANSI C++ string class; ○ For code written in Objective-C, use the NSString class; • Do not use unsafe string-handling functions that contribute to buffer overflows. GSK applications must never use the following “unsafe” functions: <ul style="list-style-type: none"> ○ strcat, strcpy, strncat, strncpy, sprintf, vsprintf, and gets. • Secure options include: <ul style="list-style-type: none"> ○ Strlcat, strlcpy, strlcat, strlcpy, snprintf, vsnprintf, and fgets • Calculate the size of all buffers and never assume that only data of a specified type and length will be received as input. In addition to standard string and integer inputs, applications must also validate the size and type of other possible inputs such as file names and media files. • Define valid format “specifiers” for functions that are susceptible to format string attacks. Refer to Apple’s Secure Coding Guide for a list of C, Carbon, and Cocoa functions that take formatted string arguments.
Exemptions:	Android applications that do not make use of the Android Native Development Kit (NDK) or contain components written in native code (C or C++)
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Buffer overflows typically occur when more data is allocated to a fixed size buffer than expected, allowing an attacker to crash the program, take control of process memory and potentially execute hostile code. These types of vulnerabilities result from a lack of proper input validation and bounds checking, and have traditionally targeted insecure C string functions. Since iOS applications are written in Objective-C, a superset of the C language, it is still possible for iOS applications to use insecure string functions such as strcpy() and strcat() that can leave iOS applications vulnerable.</p> <p>Similar to buffer overflows, integer overflow conditions occur when the computed size</p>

	<p>of an integer exceeds the allocated storage space. This type of flaw can occur when the programmer assumes the integer will only contain positive values and an unexpected negative value exceeds the intended storage space. This condition also occurs in functions used to compute the allocated space for a buffer, in which expressions may produce smaller than expected sizes that can introduce buffer overflows when the buffer is later accessed.</p> <p>Format string vulnerabilities occur when programmers do not specify how user-supplied data should be formatted, allowing attackers to supply custom formatted strings that can crash the program, leak sensitive data from memory, or potentially write to the program's stack memory. There are several standard C functions that are vulnerable to format string attacks, as well as Carbon and Cocoa Foundation functions commonly used in iOS applications.</p> <p>While vulnerabilities such as buffer overflows and format string attacks are far more pervasive in native code such as C and Objective-C, it is still possible, albeit unlikely for these issues to pose a serious risk in Android applications written in Java. This is due largely to the automatic bounds checking and memory management performed by Java. However, Android applications that use native components through the Java Native Interface (JNI) are still susceptible to the same risks described here. Although the use of JNI is not common, some Android applications make use of shared libraries written in C/C++ that can increase their risk for these types of vulnerabilities.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> • OWASP Mobile Top 10 – Client Side Injection - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project • <u>Apple Documentation - Buffer and Integer Overflow Attacks</u> • http://developer.apple.com/library/mac/#documentation/security/conceptual/SecureCodingGuide/Articles/BufferOverflows.html <u>Apple Documentation - Format String Attacks</u> <p>http://developer.apple.com/library/mac/#documentation/security/conceptual/SecureCodingGuide/Articles/ValidatingInput.html</p>
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>iOS applications should use higher level APIs available in Objective-C and use Cocoa objects such as the NSString class, providing better protection against attacks such as buffer overflows. If there is a need to use C code in the application, safer string manipulation functions should be used such as the strl* family of functions. In the</p>

case of integer overflows, Objective-C objects such as NSInteger do not provide any additional protection, and additional measures must be taken to verify computed integer values and buffer sizes fall within expected ranges.

To protect against format string attacks, verify that all format string functions accepting untrusted input define valid format “specifiers.” This is especially critical for standard C functions in which the “%n” format string can be supplied, allowing write access to memory and possible code execution.

Format String Examples:

Insecure usage with standard C function:

```
printf(buffer)
```

Correct usage with standard C function:

```
printf("%s", buffer)
```

Insecure usage with Objective-C function:

```
NSLog(message);
```

Correct usage with Object-C function:



```
NSLog(@"%@", message);
```

Android Guidelines:

Android application components written in native code (C or C++) using the Android Native Development Kit (NDK) must follow the same guidelines above.



5.1.7. Mobile applications that utilise cookies must implement them securely

Description:	<p>Mobile applications which use cookie-based authentication for session management (most commonly Web/HTML5 applications), must enable secure cookie properties using the built-in session management capabilities of the mobile development platform.</p> <p>For mobile applications that implement their own web browser components using the WebKit browser engine, applications must use built-in cookie manager classes of the mobile platform to configure the following cookie properties:</p> <ul style="list-style-type: none"> • Set “path” and “domain” values to the most restrictive settings feasible for the application; • Use non-persistent cookies. The mobile application should use “session-only” cookies that are invalidated when the user’s session is closed (e.g. when
---------------------	--

	<p>closing the mobile application);</p> <ul style="list-style-type: none"> • Ensure that session IDs contain only random values with high entropy (typically a standard feature of built-in cookie management classes); • Use the “Secure” cookie flag to force the cookie to be sent over HTTPS transport only; • Use the “HttpOnly” flag if available for the given mobile platform to help protect against malicious script access and XSS attacks. Currently the “HttpOnly” flag is not well supported for popular mobile web browsers.
Exemptions:	Persistent cookies used for approved Company analytics services.
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	Mobile applications that implement traditional browser-based components and cookie authentication must implement secure session management features in order to prevent attacks against vulnerabilities.
See Also:	<p>External References:</p> <ul style="list-style-type: none"> • OWASP Mobile Top 10 – Improper Session Handling, Poor Authorization and Authentication - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project • http://developer.apple.com/library/mac/#documentation/Cocoa/Reference/Foundation/Classes/NSHTTPCookie_Class/Reference/Reference.html • http://developer.android.com/reference/org/apache/http/cookie/package-summary.html • http://developer.android.com/reference/android/webkit/CookieManager.html
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>Leverage the NSHTTPCookie and NSHTTPCookieStorage classes to manage cookie properties, including standard mechanisms to transmit cookies to web servers through HTTP requests.</p>

	<p><u><i>Android Guidelines:</i></u></p> <p>Leverage the Android org.apache.http.cookie.Cookie and android.webkit.CookieManager classes to manage cookie properties, including standard mechanisms to transmit cookies to web servers within HTTP requests.</p>
--	---



5.1.8. Applications must follow secure memory management practices

Description:	<p>For applications targeting mobile platforms such as iOS, explicit memory management must be handled by the developer and standard practices must be followed to avoid stability and security issues. On platforms such as Android, in which applications are primarily written in Java, memory management is largely handled by the JVM through automatic garbage collection.</p> <p>The memory management issues that must be addressed include:</p> <ul style="list-style-type: none"> • Limit creation of objects in memory that hold user credentials and encryption keys, and release or de-allocate these from memory after application use; • Properly release objects that are no longer referenced to avoid memory leaks; • Do not releasing or overwrite memory that is still in use, potentially causing the program to crash or corrupt memory; • Properly de-allocated objects from memory to prevent malicious code from overwriting released memory buffers; • Avoid double free conditions, in which multiple calls are made to release the same memory block.
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Mobile applications must follow good memory management practices to prevent security and stability issues.</p> <p>It is expected behavior that sensitive data will reside in memory while the application is in use (e.g. user is authenticated). Once the user signs out or the application is closed</p>

	or put in a suspended state, the application must take additional measures to remove highly sensitive data from memory.
See Also:	<p>External References:</p> <ul style="list-style-type: none"> • http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/MemoryMgmt/Articles/MemoryMgmt.html • http://developer.apple.com/library/ios/#documentation/cocoa/conceptual/memorymgmt/Articles/mmPractical.html
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>For iOS applications, developers must explicitly manage “retain” and “release” methods in code to properly release objects from memory. When using Cocoa/Objective-C classes such as NSObject and NSString, objects can be declared as “auto-release” types in which object memory allocations and deallocations are performed behind the scenes by the compiler.</p> <p>When using Cocoa/Objective-C developers should avoid explicit calls to malloc() and dealloc() functions, and use higher-level Objective-C classes and methods for safer memory management.</p> <p>Leverage tools such as the Clang Static Analyzer within Xcode and Instruments to help detect issues such as memory leaks and object deallocation issues.</p> <p><u>Android Guidelines:</u></p> <p>When developing in Android, tools available within the Android SDK such as DDMS and the Memory Allocation Tracker can help identify memory management issues.</p> <p>It is also possible to directly invoke the Android Garbage Collector using System.gc(). While this is not always a recommended practice, this method can be useful to clean up memory so that the application cleans up large amounts of objects such as media files. Another possible use of explicit garbage collection is to purge sensitive objects from program memory upon exiting the application.</p>



5.1.9. Applications must not include hard-coded credentials and keys in application binaries and source code

Description:	GSK applications must not package hard-coded credentials that can be used to gain
---------------------	---

	<p>unauthorized access to the mobile application and server-side resources.</p> <p>The following must not be present in application packages, binaries or application property files within mobile applications:</p> <ul style="list-style-type: none"> • Service accounts used to authenticate to backend services and third party resources; • Authentication tokens and shared secrets used for authentication and authorization to backend services and web servers; • Hard-coded application user accounts and passwords left over in property and configuration files from testing and development activities; • Hard-coded test credentials and authorization tokens (e.g. OAuth key pairs) left over in source code; • Credentials used to access services on the mobile OS, such as Android Account Authenticator services used to access online resources on behalf of the user; • Passwords and constants intended to protect private application data, such as passwords used to decrypt key stores on the device
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Given the ease with which iOS and Android applications can be disassembled and reverse engineered, an attacker could access hard-coded credentials and use them to gain unauthorized access to the application and server-side resources. Since hard-coded credentials are commonly used to ease integration with web services and third parties, such credentials could potentially be abused to compromise the entire mobile application architecture or cause denial-of-service on middleware and server-side resources that are typically only accessed through mobile application clients.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> • OWASP Mobile Top 10 – Sensitive Information Disclosure, Poor Authorization and Authentication, Broken Cryptography - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

Examples:



5.1.10. Applications must secure copy and paste functions

Description:	<p>Do not rely on the operating system “Clipboard” or “Pasteboard” features as secure mechanism to store or transfer data. If the application uses a “private” Clipboard as a temporary data store, verify that contents are not accessible globally on the device or to other third party applications. If there is a strong business case to use Clipboard functions, verify that the application properly removes Clipboard data immediately after use or upon application exit.</p> <p>When working with Highly Confidential GSK data, disable the copy and paste menu from text-based UI views. It is also possible to selectively disable functions in a menu where applicable (e.g. Copy, Paste) and allow only required methods (e.g. Select). In addition to text-based menus, consider also disabling copy and paste functions when rendering sensitive data (e.g. attachments, HTML) in embedded web views.</p>
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Applications that make use of Clipboards can leave sensitive data exposed in publicly-accessible locations on the device. Additionally, users must not have the ability to extract sensitive application data from menus using copy and paste functions.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 – Sensitive Information Disclosure - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project http://developer.apple.com/library/ios/#documentation/StringsTextFonts/Conceptual/TextAndWebiPhoneOS/UsingCopy,Cut,andPasteOperations/UsingCopy,Cut,andPasteOperations.html http://developer.apple.com/library/ios/#DOCUMENTATION/UIKit/Reference/UIPasteboard_Class/Reference.html

	<ul style="list-style-type: none"> • http://stackoverflow.com/questions/1426731/how-disable-copy-cut-select-select-all-in-uitextview • http://stackoverflow.com/questions/5995210/disabling-user-selection-in-uiwebview • http://developer.android.com/guide/topics/clipboard/copy-paste.html
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>Sample code for clearing the iOS “General” Pasteboard:</p> <pre>[UIPasteboard generalPasteboard].string = nil;</pre> <p>Invalidate a private application pasteboard:</p> <pre>+ (void)removePasteboardWithName:(NSString *)pasteboardName</pre> <p>Clear pasteboard items within the applicationWillTerminate delegate:</p> <pre>pasteBoard.items = nil</pre> <p>Sample code for disabling the copy and paste menu in a UITextView object:</p> <pre>-(BOOL)canPerformAction:(SEL)action withSender:(id)sender { [UIMenuController sharedMenuController].menuVisible = NO; return NO; }</pre> <p>Sample code for disabling select/copy in a UIWebView object:</p> <pre><style type="text/css"> * { -webkit-user-select: none; } </style></pre> <p><u>Android Guidelines:</u></p> <p>For Android applications that make use of the Clipboard Framework and</p>

	ClipboardManager class, do not place sensitive application data on the system-wide clipboard.
--	---



5.1.11. Applications must build SQL queries use prepared statements and parameterized and validated

Description:	<p>GSK mobile applications that accept data from untrusted sources must protect against client-side injection attacks. Similar to traditional web applications, mobile applications that use SQLite databases for storage can be susceptible to SQL injection attacks, in which data enters the application from an untrusted source and is used to construct dynamic SQL queries.</p> <p>Client-side injection can also occur with other data source types such as XML documents, in which unvalidated input supplied to an XML query is used to inject data or retrieve contents from XML-based services.</p> <p>GSK applications must construct SQL queries safely using prepared statements with parameterized values passed into the statements. All inputs for parameters used to build SQL statements must be positively validated for type, format, and allowed characters, and minimum/maximum field length.</p>
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	SQL injection and XML injection vulnerabilities in mobile applications could allow unauthorized access or modification to records, or allow an attacker to manipulate business logic and intended security flow within the application.
See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 – Client Side Injection - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project http://www.sqlite.org/c3ref/stmt.html

	<ul style="list-style-type: none"> • http://www.sqlite.org/c3ref/prepare.html
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>When working with SQLite databases in iOS applications, leverage the sqlite_prepare_v2 or sqlite_prepare16_v2 functions to create prepared SQL statements, and use sqlite3_bind_*() interfaces to bind variables to explicit types before running SQL queries.</p> <p>Sample code using prepared statement in iOS:</p> <pre>sqlite3_stmt* selectStatement; const char *sql = "SELECT a,b,c FROM table where id = ?"; sqlite3_prepare_v2(db, sql, -1, &selectStatement, NULL); sqlite3_bind_int(selectStatement, 1, id); int eval = sqlite3_step(selectStatement); sqlite3_reset(selectStatement);</pre> <p><u>Android Guidelines:</u></p> <p>When working with SQLite databases in Android applications, leverage the database.compileStatement() method to create prepared SQL statements and the bindString() and bindLong() methods to bind values to parameters before running SQL statements.</p>

5.1.12. Applications must manage password credentials and authentication data securely

Description:	<p>GSK mobile applications must adhere to the following when managing authentication credentials:</p> <ol style="list-style-type: none"> 1. Use authorization tokens in lieu of passwords stored on the device to provide more granular control and reduced impact if compromised. In this scenario, users are only required to authenticate with their username/password initially, and then they transparently receive an authorization token that is used in subsequent requests for protected resources. <p>The benefits of this design include:</p> <ol style="list-style-type: none"> a. Username and password do not need to be retained on the device b. Token expiration and revocation can be managed more efficiently on the
---------------------	--

	<p>server side. For example, open standards such as the OAuth authorization protocol can be used to implement such a design</p> <p>c. The user experience may be improved because re-authentication using their password can be made less frequent</p>
Exemptions:	<p>Passwords where stored using encryption or hashing mechanisms following implementation guidelines below. Algorithms must be compliant with the GSK Global Cryptographic Standard.</p>
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>While most mobile applications have the requirement to manage some form of user credentials on the device, failure to adequately protect user credentials is one of the most prevalent issues in mobile applications. Insufficiently protected credentials stored on devices by mobile applications present a significant risk as unauthorized access to a device can result in direct compromise of the application and other resources that share the same credentials.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 – Insecure Data Storage , Poor Authorization and Authentication - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project http://developer.apple.com/library/ios/#documentation/Security/Reference/keychainservices/Reference/reference.html http://software-security.sans.org/blog/2011/01/05/using-keychain-to-store-passwords-ios-iphone-ipad http://developer.android.com/reference/java/security/KeyStore.html
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>Never store sensitive data using the NSUserDefaults class as these properties are stored in clear-text in a local SQLite database.</p>

```
if(![password isEqualToString:@""])
    [[NSUserDefaults standardUserDefaults] setObject:password
    forKey:@"user_password_preference"];
```

When using Apple Keychain services to store credentials on the device it is possible to define accessibility settings for Keychain items, and control whether Keychain items can be migrated to another device. When adding sensitive credentials to the Keychain a protection class of `kSecAttrAccessibleAlways`, `kSecAttrAccessibleAfterFirstUnlock`, and `kSecAttrAccessibleAlwaysThisDeviceOnly` should never be used as these protection settings make Keychain secrets accessible even when the device is locked. Applications that make use of the Keychain to store credentials should use the most restrictive protection class available, `kSecAttrAccessibleWhenUnlockedThisDeviceOnly`, which makes Keychain items inaccessible when the device is locked and does not allow migration to other devices.

```
NSMutableDictionary *query = [NSMutableDictionary dictionary];
[query setObject:(id)kSecClassGenericPassword forKey:(id)kSecClass];
[query setObject:account forKey:(id)kSecAttrAccount];
[query setObject:(id)kSecAttrAccessibleWhenUnlockedThisDeviceOnly
forKey:(id)kSecAttrAccessible];
[query setObject:[inputString dataUsingEncoding:NSUTF8StringEncoding]
forKey:(id)kSecValueData];
```

```
OSStatus error = SecItemAdd((CFDictionaryRef)query, NULL);
```



Note: In the event of a lost device that is Jailbroken by an attacker, it is still possible to bypass the device's passcode protection and gain access to all Keychain items on the device. As a result, applications that handle Confidential or Highly Confidential GSK data must not rely solely on the Data Protection API for sufficient data protection. While it does provide additional protection measures, more advanced security controls may be needed, such as jailbreak detection and remote wipe capabilities, to provide reasonable assurance that highly sensitive data is protected at rest.

Android Guidelines:



When there is a need to manage Android credentials on the device, one options is to leverage the standard Java KeyStore class available within the `java.security` package to manage cryptographic secrets. When using a KeyStore to protect user authentication data, the application must take additional measures to protect the KeyStore from unauthorized access, including password-protecting the KeyStore and preferably using a Password-based encryption (PBE) method to generate unique keys to encrypt and

	decrypt the KeyStore.
--	-----------------------

5.1.13. Applications must terminate sessions and remove authentication data in a secure manner

Description:	<p>Applications must securely terminate user sessions and remove cached authentication data after the user explicitly signs out. Cached authentication data includes session cookies, authentication tokens, and password hashes that are retained on the device for the duration of the user's session.</p> <p>GSK applications must implement the following guidelines to reduce the exposure of cached credentials on the device:</p> <ul style="list-style-type: none"> • Cached authentication data must be removed from device storage immediately after the user signs out or application session timeout occurs, and properly expired/invalidated on the server-side. • Applications must provide the ability for users to explicitly sign out and revoke their sessions.
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>An attacker that succeeds in compromising the mobile device physically or with remote malware can attempt to recover active user credentials and replay them to gain unauthorized access to GSK applications.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> • OWASP Mobile Top 10 – Insecure Data Storage , Poor Authorization and Authentication - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
Examples:	

5.1.14. Applications must implement strong transport layer encryption and strict SSL/TLS validation

Description:	<p>The mobile client application must implement strong transport layer security to encrypt communications between the mobile application and server-side components. The application must utilize a minimum of SSLv3/TLSv1 for all sensitive communications, including the login sequence and all subsequent requests and responses between client and server containing sensitive authentication and application data.</p> <p>Additionally, the application must implement strict SSL validation to negotiate a secure session between client and server. The mobile client application should be designed to fail closed if SSL validation errors occur on either the client or server side, and the ability to bypass SSL verification must be disabled for all production releases. In contrast to traditional web and mobile browser applications in which users can click through invalid certificate warnings, native and hybrid mobile applications have the opportunity to disallow use of the application when an SSL validation error occurs, regardless of the end user's decision.</p> <p>For mobile applications that use non-standard HTTP protocols, or communicate using sockets and stream-based protocols, the application must still transmit requests over secure transport channels such as HTTPS, SSH, and Secure FTP.</p>
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Applications that do not transmit network requests securely can fall victim to man-in-the-middle attacks, in which an attacker could perform unauthorized eavesdropping and intercept personal information and Confidential GSK data. This risk is especially prevalent with mobile devices, in which users are more likely to use untrusted public networks.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 - Insufficient Transport Layer Protection - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project http://software-security.sans.org/blog/2011/01/07/secure-coding-iphone-ipad-

	<p>apps-mitm-2/</p> <ul style="list-style-type: none"> • http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/Streams/Articles/NetworkStreams.html#//apple_ref/doc/uid/20002277-BCIDFCDI • http://developer.android.com/reference/org/apache/http/package-summary.html • http://developer.android.com/reference/javax/net/ssl/package-summary.html
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>When initiating HTTP requests using the <code>NSURLConnection</code> class, the application must not override the default SSL verification checks through delegation, define a new category method to bypass checks, or use the private method <code>setAllowsAnyHTTPSCertificate</code> to bypass SSL validation.</p> <p>Sample Bad SSL Practice #1:</p> <pre>[NSURLRequest setAllowsAnyHTTPSCertificate:YES forHost:[webserverURL host]];</pre> <p>Sample Bad SSL Practice #2:</p> <pre>@implementation NSURLRequest (IgnoreSSL) + (BOOL)allowsAnyHTTPSCertificateForHost:(NSString *)host { return YES; } @end</pre> <p>Sample Bad SSL Practice #3:</p> <pre>- (void)connection:(NSURLConnection *)connection didReceiveAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge { if ([challenge.protectionSpace.authenticationMethod isEqualToString:NSURLAuthenticationMethodServerTrust]) if ([trustedHosts containsObject:challenge.protectionSpace.host]) [challenge.sender useCredential:[NSURLCredential credentialForTrust:challenge.protectionSpace.serverTrust] forAuthenticationChallenge:challenge]; [challenge.sender continueWithoutCredentialForAuthenticationChallenge:challenge]; }</pre>

Although proper handling of SSL verification is enabled by default for iOS applications, this feature is commonly disabled during development and testing in order to use self-signed certificates. As a result, care should be taken to ensure that SSL verification is not bypassed in production code.

When creating network socket connections using the `NSURLSession` class, establish a secure connection to the remote host by setting the `NSURLSessionSocketSecurityLevelTLSv1` property for the `NSURLSessionSocketSecurityLevelKey` key.

```
[iStream setProperty:NSStreamSocketSecurityLevelTLSv1
forKey:NSStreamSocketSecurityLevelKey];
```

Android Guidelines:

To establish secure SSL/TLS communication for Android applications use either the built-in `javax.net.ssl` package (which uses a combination of BouncyCastle and openssl open source libraries). Another alternative, based on the needs of the application or developer preferences is to use the standard Java `org.apache.http` package.



5.1.15. GeoL services must be used in a secure manner

Description:

Applications that make use of geolocation data and Location Services must avoid the use of granular location data and use the least degree of accuracy possible. Mobile platforms provide several mechanisms to track user location with varying degrees of accuracy, using GPS and other native APIs such as Google Maps. If geolocation data is inappropriately collected and stored, the application could potentially expose granular user location data to unauthorized sources.

GSK applications that require the use of location services must ensure the following:

- 1) location data is not retained in client-side log files; that
- 2) location data stored server-side is encrypted at rest, regularly purged and anonymized from individual users and mobile devices.
- 3) Applications must not continually track the user's location within the application and only provide location-aware services when needed for a requested operation.
- 4) Prompt the user to accept or 'opt-in' to geolocation during each launch of the

	application and provide a visible option within the interface to turn off the function.
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	Geolocation data must be considered sensitive and sufficiently protected from unauthorized use. GSK applications must use the least degree of accuracy required for the application and ensure that location data is not insecurely cached on the device or server-side.
See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 – Sensitive Information Disclosure, Side Channel Data Leakage - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project http://developer.apple.com/library/ios/#documentation/CoreLocation/Reference/CLLocationManager_Class/CLLocationManager/CLLocationManager.html#//apple_ref/occ/cl/CLLocationManager http://developer.android.com/guide/topics/location/obtaining-user-location.html
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>When creating location-aware iOS applications, set the desiredAccuracy property of the CLLocationManager class to the least accurate location setting needed. For example, avoid use of the kCLLocationAccuracyBest setting in favor of a less specific location setting such as kCLLocationAccuracyThreeKilometer.</p> <p><u>Android Guidelines:</u></p> <p>When working with Location Services in Android applications, avoid requesting the ACCESS_FINE_LOCATION permission unless a strong business case exists. For applications that require location services, preferably make use of the ACCESS_COARSE_LOCATION permission for less granular location tracking.</p> <p><manifest ... ></p>

	<pre><uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" /> ... </manifest></pre>
--	---

Effective

Examples:

Implementation:

Android Guidelines:

The following is a sample custom permission declared in an AndroidManifest.xml file. In this example the “protectionLevel” attribute is set to “2” which is equivalent to the “signature” setting.

```
<permission
    android:label="@string/someLabel"
    android:description="@string/someDescription"
    android:name="com.example.android.PRIVATE_ACTIVITY"
    android:protectionLevel="2"
>
</permission>
```

Sample Activity component with custom permission:

```
<activity
    android:name="com.example.android.ChangePasswordActivity"
    android:permission="com.example.android.PRIVATE_ACTIVITY"
    android:exported="false"
>
```

Sample Content Provider component with custom permission:



```
<provider
    android:name="com.example.android.myFileProvider"
    android:permission="com.example.android.PRIVATE_ACTIVITY"
    android:exported="false"
    android:authorities="com.example.android.myFileProvider"
>
</provider>
```

Sample permission using the “permissionGroup” attribute from Android’s SDK documentation:

```
<permission android:name="com.me.app.myapp.permission.DEADLY_ACTIVITY"
    android:label="@string/permlab_deadlyActivity"
    android:description="@string/permdesc_deadlyActivity"
    android:permissionGroup="android.permission-group.COST_MONEY"
    android:protectionLevel="dangerous" />
...
</manifest>
```



5.1.16. Applications must perform strong input and output validation for all data received by the application and rendered to the user

Description:	<p>GSK mobile applications must ensure that all data received and processed by the application is sufficiently validated, including data accepted from third-party sources. The developer must identify the various entry points of data into the application and define a set of “known good” values for each possible entry point.</p> <p>All GSK mobile applications must follow a consistent approach to input and output validation, including:</p> <ul style="list-style-type: none"> • Enforce data validation on both the client and server side; • Apply a whitelist approach to allow only “known good” values; • Validate that special and metacharacters with special meaning on the target platform have been removed or properly escaped (such as %, <, >, (,), /, , !, &, *, {, }). • Validate all content types (e.g. string, integer), minimum/maximum lengths, and correct encoding types. • Perform canonicalization of data to reduce it to its simplest form. This measure can help to protect against attacks that use additional encoding techniques to bypass filters; • Validate XML input data against a valid XML Schema; • Validate user-supplied file inputs, which could allow an attacker to control file system path arguments. <p>In addition to validating all input data received by the application, GSK mobile applications must apply protections to data the application will output, including:</p> <ul style="list-style-type: none"> • Encode data using URL or HTML encoding, helping to protect against common attacks such as Cross Site Scripting; • Define the correct content encoding method for the mobile browser, which can be specified HTTP header responses, e.g., <ul style="list-style-type: none"> - Content-Type: text/html; charset=utf-8 - Content-Type: application/json
Exemptions:	

Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Mobile applications are subject to many of the same input and output validation attacks as web applications, since they typically communicate over HTTP and make heavy use of web services. In some cases the risks posed by traditional web vulnerabilities may be nullified or less impactful if a mobile application uses non-standard communications (e.g. binary protocol over an encrypted channel), or the client interface does not have features such as JavaScript enabled. However, attackers can use the same “fuzzing” techniques to send your application unexpected inputs in attempt to discover common vulnerabilities such as Cross Site Scripting and logic flaws in your mobile application if input and output validation is not performed in a consistent manner.</p> <p>Examples of common entry points include:</p> <ul style="list-style-type: none"> - HTTP requests and responses, including all HTTP parameters, headers, and cookies; - Web service requests and responses, including SOAP XML-based services and RESTful APIs, which often communicate over standard HTTP(S); - URLs invoked from untrusted sources, especially relevant when mobile applications implement custom URL/URI handlers; - XML files accepted from untrusted sources; - File I/O operations accepted from user-controlled input, in which attackers can manipulate the file system path to read/write arbitrary files or alter program behavior; - Standard user input fields in client interfaces, which may be sent in network requests for server-side processing or stored in local files and databases on the device; - User-supplied input that is processed by the application and rendered in a mobile UI component such as a web view.
See Also:	<p>External References:</p> <ul style="list-style-type: none"> • OWASP Mobile Top 10 – Weak Server Side Controls, Client Side Injection, Security Decisions Via Untrusted Inputs - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project • https://www.owasp.org/index.php/Data_Validation (Concepts also apply to mobile



	<p>applications)</p> <ul style="list-style-type: none">• https://www.owasp.org/index.php/Input_Validation_Cheat_Sheet• http://developer.apple.com/library/ios/#DOCUMENTATION/Security/Conceptual/SecureCodingGuide/Articles/ValidatingInput.html
Examples:	<p><u>Implementation:</u></p> <p>Refer to the OWASP project for sample input validation techniques using whitelisting and encoding methods.</p>

5.1.17. Applications must avoid detailed error messages that are useful to an attacker

Description:	<p>GSK mobile applications must implement secure error handling to reduce the information disclosed to a malicious party in both client and server-side components. When an unhandled exception occurs within the application, it could result in detailed information that reveals the inner workings of the application.</p> <p>GSK mobile applications must follow these guidelines when designing error and exception handling schemes:</p> <ul style="list-style-type: none"> • Provide generic error messages whenever possible, and do not display system information such as stack traces or the exact reason why an error occurred when dealing with security-relevant logic. • Define a standard approach for exception handling within the application, to reduce the risk of information disclosure through error messages. • Remove verbose logging statements in source code for production release applications. GSK applications must never include sensitive authentication data and parameter values within logging statements, including clear-text usernames/passwords, password hashes, and values such as encryption keys.
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Lack of proper error and exception handling within GSK mobile applications can disclose Confidential or Highly Confidential GSK data to an attacker wishing to compromise the application and mobile service infrastructure.</p> <p>Information disclosure flaws commonly occur under the following conditions:</p> <ul style="list-style-type: none"> • Attackers force an unhandled exception in the application, divulging system information such as stack traces and detailed error messages that reveal useful information; • Verbose logging is enabled within code that can leak system information when a valid exception does occur, including detailed logging enabled within classes and methods used to control security functions and business logic (e.g. encryption/decryption, authentication/authorization, network requests and



	<p>service API calls)</p> <ul style="list-style-type: none">• Invalid input is submitted in a network request (HTTP request or web service call) and returns a detailed error message to the user. Some common examples of this are detailed messages returned from a login service that indicate whether the supplied user is valid, and detailed SQL/database errors returned to the client.• When communicating with a web server or web service the exact software version or service banner is returned.• Other security weaknesses, such as debugging mode left enabled in production releases, amplify the impact of insecure logging statements. For example, the usual runtime of a mobile application may allow users to see critical “Warning” and “Error” level messages, while a debugging mode application will disclose “Debug” level statements in code during the application’s execution. This can mean the difference between attackers learning the name of a function called, versus the entire function call with actual input values passed to the function.
See Also:	<p>External References:</p> <ul style="list-style-type: none">• OWASP Mobile Top 10 – Sensitive Information Disclosure - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
Examples:	

5.1.18. Select an appropriate-strength authentication process in accordance with Appendix A.



Description:	Select an authentication option that is appropriate to the threat level, criticality, and sensitivity of the information. See overview of authentication methods and selection criteria in Appendix A below.
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input type="checkbox"/>, Native: <input type="checkbox"/>, Web/HTML5: <input type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input type="checkbox"/>, High <input type="checkbox"/></p>
Rationale:	Prevent various scenarios of compromise due to deployment of authentication process that is inadequate to address risk level.
See Also:	
Examples:	A majority of applications will be able to achieve compliance by externalizing authentication to GSK standard access controls that authenticate against the V-SED or Active Directory (directly, via Windows Integrated Authentication or through GSK Single Sign On)

5.1.19. Applications must enforce authorization on all protected resources

Description:	<p>GSK mobile applications must take several measures to ensure that client and server-side resources are sufficiently protected from unauthorized access. This includes both local files and resources on the device and server-side resources that mobile applications communicate with.</p> <p>As part of the application's authorization design, all GSK applications must implement the following principles:</p> <ol style="list-style-type: none"> 1) Require authentication for all resource access attempts. 2) Verify the presence of authentication data (session IDs, SSO tickets, custom authorization tokens) for all subsequent requests for application resources and web service API calls. 3) When an authentication or authorization decision fails or an unhandled
---------------------	---


	<p>exception occurs, the application must fail closed and deny access;</p> <ol style="list-style-type: none"> 4) Enforce proper role-based access control to all application-critical functions, data, and URLs. 5) Authorizations must be enforced using a centralized server-side mechanism. GSK mobile applications must never rely on client-side logic and settings to make access control decisions. 6) Mobile applications that contain highly privileged functionality, such as administrative pages, must not make such resources accessible to general users over Internet. 7) Service accounts used to authenticate with server-side resources on behalf of the user must be configured with minimal privileges. For example, a service account used on behalf of the user to write files to a web server must not be granted read/write access to all resources on the server.
Exemptions:	Publicly-accessible portions of mobile applications and server-side resources (e.g. images) that do not store or process Confidential GSK data
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Mobile application resources, including the resources consumed from web servers and web services, must enforce strong authorization requirements to protect Confidential GSK data and applications from unauthorized access. This includes protecting resources from anonymous access, in addition to privilege escalation attacks by authenticated users.</p> <ol style="list-style-type: none"> 1) For example, an attacker may be able to tamper with client-side configuration settings and capture inbound network responses to modify intended security flows.
See Also:	<p>External References:</p> <ul style="list-style-type: none"> • OWASP Mobile Top 10 – Poor Authorization and Authentication, Weak Server Side Controls, Security Decisions Via Untrusted Inputs - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
Examples:	

5.1.20. Mobile applications must display an approved End-User License Agreement (EULA) that includes consent for the submission of PII to GSK.

Description:	<p>All externally facing mobile applications must display an End-User License Agreement (EULA) informing users of data collection and usage policies. If the globally approved EULA is modified or if an alternate EULA is used, it must be approved by local Legal Counsel prior to implementation.</p> <p>For applications that explicitly collect PII and submit this information back to GSK, or make use of identifiers that link to data stores containing PII, user consent must be obtained at install time or when taking actions in the application that require the transfer of PII back to GSK.</p> <p>Publicly accessible mobile applications that enable the capture of free form text that may include PII but where this information is not transmitted back to GSK need not obtain consent.</p> <p>Contact local legal counsel for the appropriate EULA.</p>
Exemptions:	
Applies To:	<p>Platforms:  </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>EULAs provide a means to inform users on the application's data collection and usage practices. The EULA also serves to obtain consent from users prior to the collection and usage of PII, which is mandatory in many regions including the European Union.</p>
See Also:	
Examples:	

5.2. Controls for iOS applications


5.2.1. Disable caching of sensitive data within Apple's dynamic dictionary log

Description:	<p>Applications that process or store Confidential or Highly Confidential GSK data must code defensively to disable the caching feature for sensitive UI screens and application fields. Caching of sensitive fields must be disabled for the following:</p> <ul style="list-style-type: none"> • Personally Identifiable Information • Account Secrets (password, PIN values, account profile information used for secondary authentication such as secret questions and answers) <p>The dynamic dictionary file is stored in the following path on the filesystem: <i>/root/Library/Keyboard/dynamic-text.dat</i></p>
Exemptions:	Mobile applications that do not present any sensitive input fields to the user.
Applies To:	<p>Platforms: </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Apple iOS devices maintain a keyboard cache of plaintext words not found in the default Apple dictionary. Keyboard input entered into third party applications is also added to this keyboard cache, including potentially sensitive application data. In the event of a compromised device, an attacker could retrieve the contents of this file and the data contained within.</p> <p>By default, data such as password fields and special characters are excluded from this log. Also, users must generally input the same keyword multiple times in an input field for it to become populated into the dynamic dictionary, helping to reduce the exposure.</p> <p>Sensitive consumer and Confidential GSK data must not be cached on mobile devices in insufficiently-protected and shared areas.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> • OWASP Mobile Top 10 – Sensitive Information Disclosure - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p>

```

UITextField* myField = [[UITextField alloc] init];
myField.autocorrectionType = UITextAutocorrectionTypeNo;
  
```


5.2.2. Disable automatic caching of iOS application screenshots

Description:	<p>Applications that handle Confidential or Highly Confidential GSK data must mask any sensitive data fields when the application is sent to the background, effectively preventing the default snapshots from exposing such data or make use of a default splash screen that is always present in the <i>/Snapshots</i> directory.</p> <p>Sensitive fields for which snapshot “backgrounding” must be disabled include:</p> <ul style="list-style-type: none"> • Personally Identifiable Information • Account Secrets (password, PIN values, account profile information used for secondary authentication such secret questions and answers) <p>Snapshots are saved in the following path within the application’s container: <i>/private/var/mobile/Applications/<APP_GUID>/Library/Caches/Snapshots/</i></p>
Exemptions:	<p>Mobile applications that do not display any GSK Confidential or Highly Confidential unmasked information in application interfaces.</p>
Applies To:	<p>Platforms: </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>By default, iOS applications capture snapshot images of the application UI when the Home key is used and when the user minimizes an opened application. This iOS feature is used to provide a smoother scaling effect when the application is re-opened, but can potentially capture sensitive data in snapshots. Failure to secure application snapshots could result in Confidential or Highly Confidential GSK data leakage if the device is lost, stolen, or changes ownership.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> • OWASP Mobile Top 10 – Sensitive Information Disclosure - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

	<ul style="list-style-type: none"> Refer to the following article and iOS Application Programming Guide for detailed guidance. http://software-security.sans.org/blog/2011/01/14/whats-in-your-ios-image-cache-backgrounding-snapshot/
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>Remove sensitive information from views before moving the application to the background. This can be accomplished by setting sensitive fields as hidden within the applicationDidEnterBackground delegate method, as shown below. It is also necessary to unhide the fields when restoring the view within the applicationDidBecomeActive delegate.</p> <pre> - (void)applicationDidEnterBackground:(UIApplication *)application { viewController.userNameField.hidden = YES; viewController.secretQuestionField.hidden = YES; viewController.secretAnswerField.hidden = YES; } </pre> <p>As an alternate approach to hiding sensitive UI fields, create a default splash screen that always overrides the default snapshot backgrounding feature.</p> <p>If the methods described above are not sufficient based on application characteristics (e.g. screen capture of sensitive image or other data), it is possible to disable the backgrounding feature entirely by setting the "<i>Application does not run in background</i>" property in the application's <i>info.plist</i> file.</p>


5.2.3. Custom URL schemes must be implemented in a secure fashion

Description:	<p>GSK iOS applications that implement custom URL schemes must ensure that:</p> <ul style="list-style-type: none"> Custom URL handler actions do not originate from untrusted sources, or the application properly handles authorization requests if such a use case exists. URL handler requests and associated parameters are properly sanitized before being accepted and processed within the application. <p>When creating Apple iOS applications it is possible to define "URL Schemes" that act as URL protocol handlers used to share data between applications. URL Schemes provide an inter-process communication (IPC) mechanism that allows actions to be invoked by other applications such as Mobile Safari. It is also possible for other third party applications to invoke custom URL handlers and potentially trigger unauthorized</p>
---------------------	--

	actions implemented by the URL scheme.
Exemptions:	
Applies To:	<p>Platforms: </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	Applications that define custom URL schemes can potentially be abused by malicious third party sources in order to execute private actions within GSK applications without the user's consent.
See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 – Security Decisions Via Untrusted Inputs - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project http://developer.apple.com/library/IOS/#documentation/UIKit/Reference/UIApplicationDelegate_Protocol/Reference/Reference.html http://software-security.sans.org/blog/2010/11/08/insecure-handling-url-schemes-apples-ios
Examples:	<p><u>Implementation:</u></p> <p><u>iOS Guidelines:</u></p> <p>iOS applications that define custom URL schemes must take extra caution to only accept URL requests from known source applications, request user authorizations to invoke actions, and properly sanitize URL requests and parameters.</p> <p>One example of such a custom URL handler is the tel: scheme that can be used to launch the Device application from HTML in a web browser. When invoking the tel: URL handler Safari launches a pop-up notification requesting authorization to make a call. However, a large number of applications that register custom URL schemes do not request user authorization when actions are requested from untrusted sources.</p> <p>The application:openURL:sourceApplication:annotation: method should be used to accept URL handler requests from only known source applications, as shown in the</p>


	<p>example below.</p> <pre>- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication annotation:(id) annotation</pre>
--	---

5.2.4. iOS applications must not use the device's Unique Identifier (UDID) for authentication purposes

Description:	By default iOS devices are assigned a Unique Identifier (UDID) that is derived from hardware information. Since it is possible to change and spoof the device's UDID, this value must not be relied upon to perform device and user authentication.
Exemptions:	
Applies To:	<p>Platforms: </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>The UDID value is commonly used by ad networks and other companies to track user behavior, and GSK applications must not make use of the UDID for tracking purposes since it is generally considered a privacy violation.</p> <p>Use of the UDID for authentication purposes could allow an attacker to easily impersonate another user, potentially allowing access to GSK mobile applications and associated data.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 – Poor Authorization and Authentication - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
Examples:	

5.2.5. Mobile applications that deliver content in embedded web pages must protect against UI spoofing attacks.


Description:	Mobile applications that implement their own web browser components must protect against user interface spoofing attacks, in which attackers can attempt to spoof the UIs
---------------------	---

	<p>of legitimate applications and trick the user into visiting malicious websites.</p> <p>GSK mobile applications that utilise web view components within the application, such as loading external URLs, must display the URL bar to help protect users from phishing and UI spoofing attacks.</p>
Exemptions:	
Applies To:	<p>Platforms: </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>If the URL bar is not present for web views within the application, an attacker could perform a man-in-the-middle attack on mobile users, detect that they are originating from mobile devices based on the User-Agent string, and spoof legitimate UI pages the user is attempting to access. Depending on the UI page impersonated, a successful spoofing attack could result in compromise of GSK Confidential or Highly Confidential data from the mobile device.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> http://software-security.sans.org/blog/2010/11/29/ui-spoofing-safari-iphone/
Examples:	

5.3. Controls for Android applications

5.3.1. Android applications must protect their components from unauthorized access and abuse


Description:	<p>GSK Android applications must protect their components from unauthorized access and abuse by third party applications and malware. Within the application's AndroidManifest.xml file, custom security permissions must be defined to protect access to private application components, including:</p> <ul style="list-style-type: none"> Activity permissions to restrict who can start an activity (e.g. launch application UI screens, including UIs that should only be accessible to authenticated users and roles)
---------------------	--

	<ul style="list-style-type: none"> - Service permissions to restrict who can start or bind to a service (e.g. service that syncs data between the application and external source) - Broadcast Receiver permissions to restrict who can send broadcast messages to receivers, and which applications can receive message intents from broadcast receivers (e.g. Receivers used to wait and respond to system events and take some action, such as the device reboot) - Content Provider permissions to restrict who can access application data published through a content provider (e.g. Provider used to share application files) <p>GSK mobile applications that process Confidential or Highly Confidential data must restrict access to these components using the following guidelines:</p> <ol style="list-style-type: none"> 1) Declare custom permissions within the AndroidManifest.xml file using one or more <permission> tags; 2) Define the <protectionLevel> attribute with a value of “signature”. 3) Enforce permissions on the desired component by including the android:permission attribute within the component’s definition in the AndroidManifest.xml file and naming the permission assigned to protect it.
Exemptions:	
Applies To:	<p>Platforms: </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Each Android application contains a manifest file which defines the Android components that comprise the application along with the permissions used to access system APIs and protect communication between application components.</p> <p>Android applications are divided into four main types of components, including Activities, Services, Broadcast Receivers, and Content Providers. The Android platform provides an inter-process communication (IPC) mechanism that facilitates communication between system and application components, and is based on sending and receiving of broadcast messages (known as Intents) between components. This design allows third party applications to receive notifications from Android system events, and also broadcast and receive message Intents with other applications.</p>

	<p>If Android applications are not sufficiently protected, components can be made accessible to other third party applications, including sensitive application functionality only intended for private use and inter-application communication. Examples of abuse scenarios with Android components include:</p> <ul style="list-style-type: none"> • Allowing other applications to launch your application’s activities, such as UI screens that should only be accessible with proper authentication and authorization; • Binding to or starting services, which could allow a malicious party to intercept or modify your application’s communications; • Allowing third party applications to send broadcast messages to your application and invoke some action (e.g. “Intent fuzzing”) • Allowing third party applications to receive broadcast messages not intended for them (e.g. unauthorized recipients); • Allowing access to data and operations available through your application’s Content Provider components. <p>The “signature” permission setting is used to grant access to components only if the requesting application is signed with the same certificate as the application that declared the permission. The “signature” setting is generally the preferred method for enforcing custom permissions unless user interaction is required.</p>
<p>See Also:</p>	<p>External References:</p> <ul style="list-style-type: none"> • OWASP Mobile Top 10 – Poor Authorization and Authentication, Security Decisions Via Untrusted Inputs - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project • http://developer.android.com/guide/topics/security/security.html • http://developer.android.com/reference/android/R.styleable.html#AndroidManifestPermission_protectionLevel • http://developer.android.com/reference/android/Manifest.permission_group.html

5.3.2. Android applications must not make their components “exportable” to other applications


<p>Description:</p>	<p>GSK Android applications must not allow their components to be exported to other applications, including Activities, Services, and Broadcast Receiver components defined in the application’s AndroidManifest.xml file.</p>
----------------------------	--

	<p>GSK applications that process Confidential or Highly Confidential information must set the “exported” attribute to false for application components defined within the AndroidManifest.xml file.</p>
Exemptions:	
Applies To:	<p>Platforms: </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>The ability for other applications on the device to access an application’s components can expose GSK Confidential data and functionality not intended for use outside of the application.</p> <p>The risk introduced by the “exported” attribute setting varies based on nature of the Android component. Components that define at least one set of <intent-filter> tags within the AndroidManifest.xml file implies that the component (e.g. Activity, Service) is intended for external use. Components that do not define any intent filters must be accessed explicitly by their exact class name, and such components are generally only used for inter-application communication.</p> <p>The default value for the “exported” attribute depends on whether the component uses “implicit” or “explicit” Intents. For “implicit” components (e.g. ones containing intent filters), the default setting for the “exported” attribute is true. For “explicit” components, the default setting for “exported” is false, helping to reduce the exposure.</p> <p>While setting the “exported” attribute helps reduce the attack surface on Android applications, using this measure alone can still allow for other applications to explicitly call components by their exact class name. The “exported” attribute must not be used as a replacement for permissions to protect components, but can be used in combination with permissions to provide better security.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 – Poor Authorization and Authentication, Security Decisions Via Untrusted Inputs - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project http://developer.android.com/guide/topics/manifest/activity-

	element.html#exported
Examples:	<p><u>Implementation:</u></p> <p><u>Android Guidelines:</u></p> <p>Insecure Example: Sample Android Service component with no restrictions. In this case custom permissions are not enforced and the “exported” flag defaults to true.</p> <pre> <service android:name="com.example.android.PublicSyncService"> <intent-filter> <action android:name="com.example.android.SERVICE"> </action> </intent-filter> </service> </pre> <p>Secure Example: Sample Android Service component that enforces custom permissions and sets the “exported” attribute to false.</p> <pre> <service android:name="com.example.android.PrivateSyncService" android:permission="com.example.android.PRIVATE_ACTIVITY" android:exported="false"> <intent-filter> <action android:name="com.example.android.SERVICE"> </action> </intent-filter> </service> </pre>


5.3.3. Android applications must not be released with the “debuggable” flag enabled

Description:	When GSK Android applications are released for production use, they must not have the “debuggable” flag enabled within the application’s AndroidManifest.xml file.
Exemptions:	

Applies To:	<p>Platforms: </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input checked="" type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>GSK applications that are released in a “debuggable” state significantly increase the amount of information available to an attacker. Failure to restrict this setting can leak GSK Confidential data from the application, disclose the inner workings of the application (including authentication/ authorization and crypto functions used), help an attacker to better understand network communications and service APIs used, and use the information divulged to target security weaknesses in the application.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 – Sensitive Information Disclosure - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project http://developer.android.com/guide/topics/manifest/application-element.html#debug
Examples:	<p><u>Implementation:</u></p> <p><u>Android Guidelines:</u></p> <p>Sample AndroidManifest.xml file with “debuggable” flag set appropriately set to false. This attribute must not be present in production releases, or its value must be explicitly set to false.</p> <pre>android:debuggable="false"</pre>


5.3.4. Android applications must use code obfuscation techniques to protect source code from disclosure

Description:	<p>Android applications must use code obfuscation techniques to prevent an attacker from easily reversing the application and obtaining Java source code. Using freely available tools, it is possible to convert Android’s Dalvik Executable binaries into readable Java source code.</p> <ul style="list-style-type: none"> The ProGuard tool within the Android SDK must be used to obfuscate any of the following: Security-relevant classes; Cryptographic methods used to protect sensitive application data; and
---------------------	---

	<ul style="list-style-type: none"> Hard-coded authentication secrets that could introduce an immediate exposure if compromised.
Exemptions:	
Applies To:	<p>Platforms: </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>A sophisticated attacker can use knowledge of the exposed source code to reverse engineer the application and identify potential security weaknesses in the code. For example, the source code could be used by an attacker in an attempt to circumvent security logic and encryption mechanisms. Failure to obfuscate Android applications can also expose clear-text credentials and other authentication data that may be present in the code.</p> <p>Since the ProGuard obfuscation process works by replacing real class names and methods with obfuscated names, using this tool requires additional steps to be taken for developers when troubleshooting published applications. For example, another tool must be used to decode obfuscated stack traces received in bug reports.</p> <p>After a thorough risk analysis of the code, it may be desirable to obfuscate only the portions of code that are deemed sensitive, rather than obfuscate the entire application. For example, Java classes used for cryptographic and authentication/authorization functions must be deemed sensitive and protected accordingly. Additionally, classes used to send and receive network requests and communicate with backend service APIs must be considered sensitive. Based on the application's risk profile, it may be acceptable to bypass obfuscation for the remainder of the code that does not divulge any useful information to an attacker.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> OWASP Mobile Top 10 – Sensitive Information Disclosure - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project http://developer.android.com/guide/developing/tools/proguard.html http://proguard.sourceforge.net/index.html#/manual/introduction.html

Examples:	<p><u>Implementation:</u></p> <p><u>Android Guidelines:</u></p> <p>Refer to the Android SDK and ProGuard documentation.</p>
------------------	---

5.3.5. Files opened or created on Android device must use secure permissions

Description:	<p>When Android applications perform operations that create or open existing files on the device, GSK applications must enforce restrictive permissions to protect local storage resources.</p> <p>GSK mobile applications must assign the MODE_PRIVATE file creation mode to restrict file access to the calling application for the following methods used when performing local file system operations:</p> <ul style="list-style-type: none"> - Context.openFileOutput() – Used to open or create local files - Context.openOrCreateDatabase() – Used to open or create local SQLite databases - Context.getSharedPreferences() – Used to read or modify contents of the preferences file <p>GSK applications must never use the MODE_WORLD_READABLE and MODE_WORLD_WRITEABLE modes for private application files.</p>
Exemptions:	
Applies To:	<p>Platforms: </p> <p>Application Type: Hybrid: <input checked="" type="checkbox"/>, Native: <input checked="" type="checkbox"/>, Web/HTML5: <input checked="" type="checkbox"/></p> <p>Application Risk: Low <input type="checkbox"/>, Medium <input checked="" type="checkbox"/>, High <input checked="" type="checkbox"/></p>
Rationale:	<p>Android applications with insecure file permission settings can allow other applications or malware on the device to read and write private application files, potentially allowing unauthorized access to the application and Confidential GSK data.</p>
See Also:	<p>External References:</p> <ul style="list-style-type: none"> • OWASP Mobile Top 10 – Insecure Data Storage, Sensitive Information

	<p>Disclosure -</p> <p>https://www.owasp.org/index.php/OWASP_Mobile_Security_Project</p> <ul style="list-style-type: none"> http://developer.android.com/reference/android/content/Context.html
Examples:	<p><u>Implementation:</u></p> <p><u>Android Guidelines:</u></p> <p>Sample code using openFileOutput() to create a private keystore used by the application:</p> <pre>FileOutputStream os = Context.openFileOutput(KEYSTORE_FILENAME, Context.MODE_PRIVATE);</pre> <p>Sample code using getSharedPreferences() to read a property from the application's shared preferences file:</p> <pre>long lastCheckInTime = this.getSharedPreferences (SHARED_PREFS_FILENAME, MODE_PRIVATE) .getLong(SHARED_PREF_LAST_VERSION_CHECK_TIME, 0); long time = new Date(lastCheckTime).getTime();</pre>

Glossary & Administration

6. Glossary

Key terms are used as defined in the Global Glossary. For the specific purpose of this document, the following additional terms and definitions apply:

Access Control / Authorisation	The process through which the privileges (i.e. appropriate need to access) of a requestor are defined, interpreted and validated by an application.
Applicability	Standards are applicable based upon the risk factors of an application for which the controls should be applied. These factors increase the inherent risk level and therefore increase the need for addressing the control. Standards are applicable to either All Mobile Applications, Critical Applications, or High Risk Applications.
Crypto-period	The time span during which a specific key is authorized for use or in which the keys for a given system or application may remain in effect.
Cryptographic Services	A GSK IT Service with hardware, software, firmware, or some combination thereof that implements cryptographic logic or processes
Database	A collection of information stored and managed by a database management system, for example Oracle.
Decode / Decrypt	To decode data is to convert encoded data back to its original format. To decrypt data is a form of decoding that requires use of a decryption algorithm and secret key to convert ciphertext back to plain text.
Decryption Algorithm	A mathematical procedure (i.e. algorithm) for transforming encrypted cipher text information to plain text information.
Digital Certificate	A set of data that uniquely identifies an entity, contains the entity's public key and possibly other information, and is digitally signed by GSK and/or a trusted party, thereby binding the public key to the entity. Additional information in the certificate could specify how the key is used and its crypto-period. Typically formatted using the X.509 standard.
Digital Signature	The result of a cryptographic transformation of data that, when properly implemented, provides the services of: <ol style="list-style-type: none">1. origin authentication2. data integrity, and

	<p>3. signer non-repudiation.</p> <p>Alternately: data associated with a Record as a result of processing the Record using PKI, which data can be used to determine: (1) whether the data was created using the Private Key that corresponds to the Public Key in the signing Entity's Digital Certificate; and (2) whether the message has been altered since the Digital Signature was associated with the Record.</p>
Encode/Encrypt	To encode data is to convert data into a given format. To encrypt data is a form of encoding that converts plaintext data to ciphertext with use of an encryption algorithm such that it cannot be accessed without knowledge of a secret key.
Encryption Algorithm	A mathematical procedure (i.e. algorithm) for transforming plain text information into meaningless cipher text.
Event Log	An electronic file of event records which is stored. The log may be stored in a file system or database table and is generated automatically by an application or system that is writing individual event records.
Exception	A non-exempt application, page or site type that must acquire a formal approval from the GSK IT Architecture council.
Exemption	A pre-approved application, page or site type that does not need to comply with a standard.
File	A collection of electronic data in a pre-defined format that is stored on a computer or other removable media. Examples are MS-Word ".doc", text ".txt" or Program ".exe" files and temporary files ".tmp" or cookies ".dat".
File System	The system that an operating system or program uses to organise and manage files. Examples are FAT, NTFS or hierarchies within them.
HTML Document, HTML Web Pages, web page	When users visit a URL, the HTML or XHTML code, scripts, images and other content that load in the user's browser. Except where noted, this term applies to the entire page, from the header to the closing </html> tag. Portlets are not responsible for compliance of other portlets or the portal framework.
Hybrid Mobile Application	Hybrid mobile applications use a combination of Web and Native components, allowing developers to use web technologies to access to native platform APIs and device features. Hybrid applications allow developers to leverage existing web development skills without having

	deep knowledge of native mobile platform development. Since Hybrid components make use of both native and web components, many traditional web application security risks and native platform-specific risks generally apply to Hybrid mobile applications.
Key	A defined set of data that is applied as an input parameter used in conjunction with a cryptographic algorithm and affects the output of that algorithm. Refers indifferently either to asymmetric or symmetric cryptographic keys.
Key Management	The activities involving the handling of cryptographic keys and other related security parameters (e.g., IVs and passwords) during the entire life cycle of the keys, including their generation, storage, establishment, entry and output, and destruction.
Native Mobile Application	Native mobile applications are platform-specific and developed using the mobile platform's Software Development Kit (SDK). Native applications may be more expensive and timely to develop, but may also deliver the best user experience and provide access to the most native platform features. Since Native applications can also implement custom web browser components, many traditional web application security risks can also apply to Native mobile applications.
One-way hash (function)	An algorithm that transforms a message to a coded representation of the message that cannot easily be reversed. The term 'hash' is often used for this.
Pass Phrase	A sequence of words or other text used to control access to a computer system, program or data. A passphrase is similar to a password in usage, but is generally longer for added security. Passphrases are often used to control both access to, and operation of, cryptographic programs and systems (e.g. Certificates). Passphrases are particularly applicable to systems that use the passphrase as an encryption key.
Plain text	Data in decrypted (human-readable) form.
REST	REST stands for “REpresentational State Transfer”. REST is a model for Web services based solely on HTTP. REST takes the view that the Web already has everything necessary for Web services, without having to add extra specifications such as SOAP. Any item can be made available (i.e. represented) at a URI, and, subject to the necessary permissions, it can be manipulated using one of the simple operations defined within HTTP (GET to retrieve information, PUT and POST to modify it, DELETE to remove

	it). Proponents of REST-based Web services argue that retaining this very simple semantic structure is the best way of preserving interoperability between all Web participants.
Secret Key	Secret key encompasses both a private key of a public/private asymmetric key pair and a symmetric key.
Shared Access Management (AM) Service	A service that executes access control or identity management functions in support of multiple target systems. Examples of technologies managed through these services include: Strategic Enterprise Directory, Extranet Directory Service, Active Directory, PAMs, Courion, Notes/Domino, Netegrity (SSO).
SOAP	SOAP stands for “Simple Object Access Protocol”. SOAP provides a simple, extensible, and rich XML messaging framework for defining higher-level application protocols offering increased interoperability in distributed, heterogeneous environments.
Source Code	References to source code should be interpreted to include items such as C/C++, Objective C, JavaScript, ASP.NET code, configuration files etc...
Transmission	A communication path that enables data to pass from one system to another. There is no intended storage of the data in transmission.
Vulnerability	A circumstance that increases the likelihood of threats materialising, i.e. control weaknesses.
Mobile Web/HTML5 Application	Mobile Web/HTML5 applications are written entirely in HTML, CSS, and JavaScript. Mobile Web applications run in standard mobile web browsers and have the most in common with traditional web applications. Mobile Web applications provide a lightweight cross-platform solution to mobile application development, but may be less powerful than Native or Hybrid applications. Additionally, HTML5 applications do not require approval or distribution from the App Store or Android Market, as they are essentially mobile-enabled web applications. As a result, some security controls/risks in this document that apply to native platform components may not be applicable to pure Web/HTML5 based applications.
XML	eXtensible Markup Language - The data tagging language of web services. XML is not so much a language as a standardized set of rules for adding structure to any form of data using a system of markup tags. Anyone can create their own markup vocabulary (called an XML Schema), and XML ensures that the structure will be intelligible to anyone else who consults the XML Schema document. More importantly, referring to an XML

	Schema enables XML-aware software to automatically manipulate the data without needing advance knowledge of the structure.
--	--

7. Administration

Approval:	IT Risk Management & Compliance Board
Owner:	Ian Wadey, Chair ITRMCB
Author:	Kathleen Zarsky, Director Enterprise Architecture
Approval Date:	03-Dec-2012
Effective Date	31-Dec-2012
Review Date	31-Dec-2014
History:	Version 1: New document

8. Waivers

Exceptions must be raised through the Risk Exception and Acceptance Process (REAP).

9. Related Documents

STD-IT-WADS	Web Application Development Standard
-------------	--------------------------------------

Appendix A: Authentication Methods – Selection Criteria

The following table describes various authentication methods and their suitability for use by GSK mobile applications.

Authentication Method	Description and associated standards	Strength Level	Suitability
External authentication platform	<p>Authentication against standard authentication/authorization middleware platform, consisting of:</p> <p>Issuance of access tokens (e.g. such as OAuth tokens) that can be used for subsequent resource requests and delegated authorization.</p> <p>May use open standards such as OAuth protocol for authentication/authorization.</p>	Can achieve adequate strength with appropriate configuration and design.	Mobile applications for consumer access to mobile applications.
Client certificates over SSL	Requires each client to obtain a certificate. Certificates are mapped to user accounts, which are used by IIS for authorising access to Web resources/services. May not be widely supported by developer tools for building clients.		<p>Appropriate for all web applications</p> <p>However, infrastructure to support is not widely available at this time. Highly critical applications with small user bases may be feasible.</p>
Token based Authentication	Requires each client to obtain a Security Token, such as a SecureID token or Smart Card. This is also a form of two-factor authentication (what you have, what you know)		<p>Appropriate for all applications</p> <p>However, infrastructure to support is not widely available at this time. Highly critical applications with small user bases may be feasible.</p>