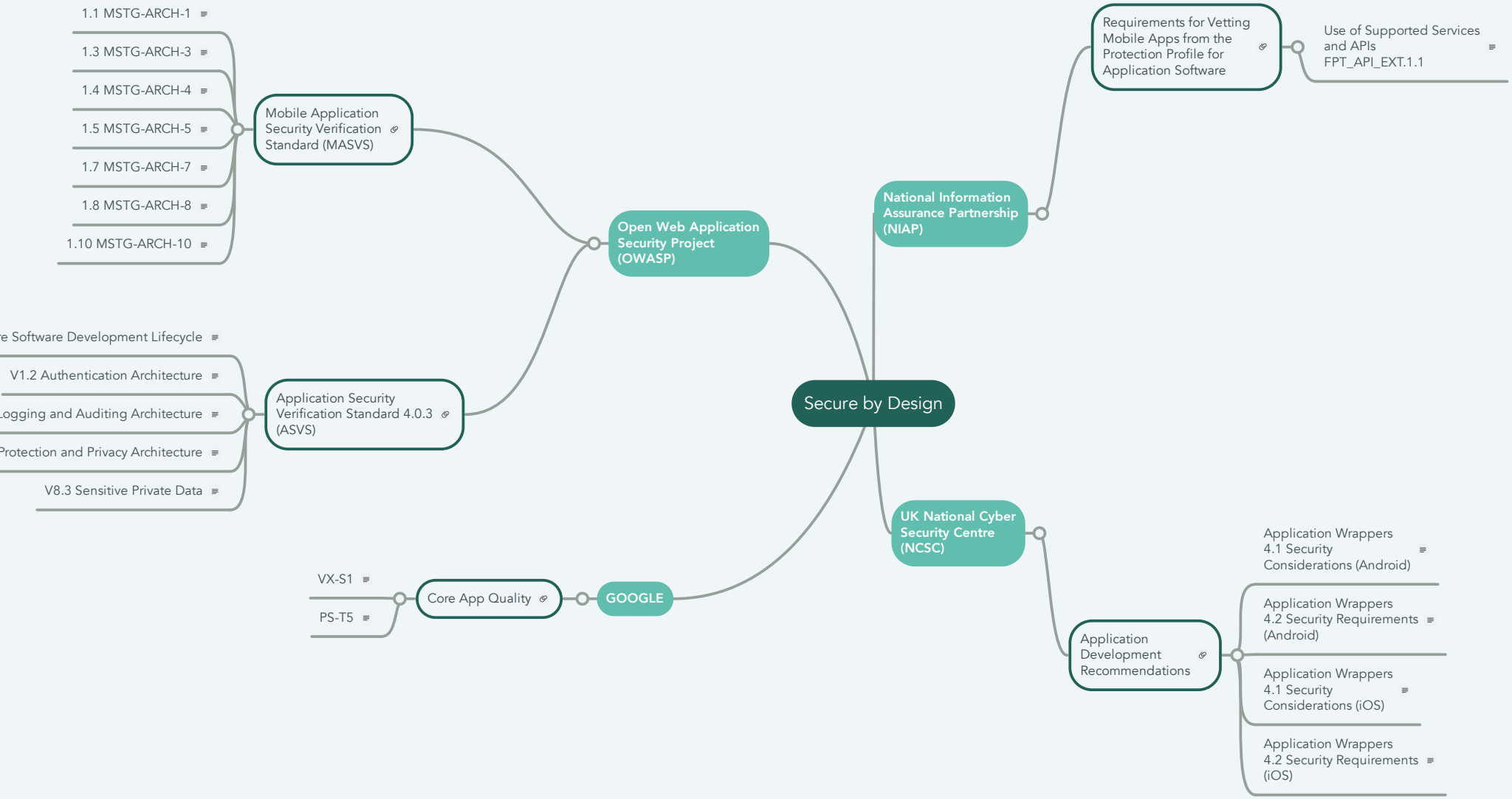
Secure by Design
Open Web Application Security Project (OWASP)
Mobile Application Security Verification Standard (MASVS)
1.1 MSTG-ARCH-1
1.3 MSTG-ARCH-3
1.4 MSTG-ARCH-4
1.5 MSTG-ARCH-5
1.7 MSTG-ARCH-7
1.8 MSTG-ARCH-8
1.10 MSTG-ARCH-10
Application Security Verification Standard 4.0.3 (ASVS)
e Software Development Lifecycle
V1.2 Authentication Architecture
Logging and Auditing Architecture
Protection and Privacy Architecture
V8.3 Sensitive Private Data
GOOGLE
Core App Quality
VX-S1
PS-T5
National Information Assurance Partnership (NIAP)
Requirements for Vetting Mobile Apps from the Protection Profile for Application Software
Use of Supported Services and APIs FPT_API_EXT.1.1
UK National Cyber Security Centre (NCSC)
Application Development Recommendations
Application Wrappers 4.1 Security Considerations (Android)
Application Wrappers 4.2 Security Requirements (Android)
Application Wrappers 4.1 Security Considerations (iOS)
Application Wrappers 4.2 Security Requirements (iOS)

Secure by Design

1. National Information Assurance Partnership (NIAP)

1.1. Requirements for Vetting Mobile Apps from the Protection Profile for Application Software

Link: https://www.niap-ccevs.org/MMO/PP/394.R/pp_app_v1.2_table-reqs.htm

1.1.1. Use of Supported Services and APIs FPT_API_EXT.1.1

> The application shall use only documented platform APIs.
> Application Note: The definition of documented may vary depending upon whether the application is provided by a third party (who relies upon documented platform APIs) or by a platform vendor who may be able to guarantee support for platform APIs.

2. UK National Cyber Security Centre (NCSC)

2.1. Application Development Recommendations

Link: https://www.ncsc.gov.uk/collection/application-development/android-application-development/application-wrappers

2.1.1. Application Wrappers 4.1 Security Considerations (Android)

> A variety of 'application wrapping' technologies exist on the market today. Whilst these technologies ostensibly come in a variety of forms which provide different end-user benefits, on most platforms (including Android) they essentially work in one of three ways.
>
> Category 1: These provide a remote view of an enterprise service, for example a Remote Desktop view of a set of internal applications that are running at a remote location, or an HTML-based web application. Multiple applications may appear to be contained within a single application container, or may live separately in multiple containers to simulate the appearance of multiple native applications. Usually, only temporary cached data and/or a credential is persistent on the device itself.
>
> Category 2: These are added to an application binary after compilation and dynamically modify the behaviour of the running application (for example to run the application within another sandbox and intercept and modify platform API calls) in an attempt to enforce data protection.
>
> Category 3: The source code to the surrogate application is modified to incorporate a Software Development Kit (SDK) provided by the technology vendor. This SDK modifies the behaviour of standard API calls to instead call the SDKs API. The developer of the surrogate application will normally need to be involved in the wrapping process.

2.1.2. Application Wrappers 4.2 Security Requirements (Android)

Category 1 technologies are essentially normal platform applications, but which store and process minimal information, deferring processing and storage to a central location. The development requirements for these applications are identical to other native platform applications. Developers should follow the guidelines given above.

Category 2 and category 3 wrapping technologies are frequently used to provide enterprise management to applications via the MDM server that the device is managed by. SDKs are integrated into these MDM solutions and can be used to configure settings in the application or to modify its behaviour. For example, the application could be modified to always encrypt all data or not use certain API calls.

On Android, both category 2 and category 3 wrapping technologies require the surrogate developer's co-operation to wrap the application into a signed package for deployment onto an Android device. As such, normally only custom developed in-house applications, and sometimes trusted third party applications (with co-operation) can use these technologies. As the robustness of these wrapping technologies cannot be asserted in the general case, they should not be used with an untrusted application; they should only be used to modify the behaviour of trusted applications, or for ease of management of the wrapped applications.

In-house applications should be developed specifically against the previously described security recommendations wherever possible. The use of app-wrapping technologies should only be used as a less favourable alternative method of meeting the given security recommendations where natively meeting them is not possible.

Ultimately, it is more challenging to gain confidence in an application whose behaviour has been modified by a category 2 technology. It is difficult to assert that dynamic application wrapping can cover all the possible ways an application may attempt to store, access and modify data. It is also difficult to make any general assertions about how any given wrapped application will behave. As such, the NCSC cannot give any assurances about category 2 technologies or wrapped applications in general, and hence cannot recommend their use as a security barrier at this time.

However, category 3 technologies are essentially an SDK or library which developers use as they would any other library or SDK. In the same way that the NCSC does not assure any standalone cryptographic libraries, we do not provide assurance in SDKs which wrap applications. The developer using the SDK should be confident of its functionality, as they would be with any other library.

2.1.3. Application Wrappers 4.1 Security Considerations (iOS)

A variety of 'application wrapping' technologies exist on the market today. Whilst these technologies ostensibly come in a variety of forms, each providing different end-user benefits, on most platforms (including iOS) they essentially work in one of three ways:

Category 1: These provide a remote view of an enterprise service. For example, a Remote Desktop view of a set of internal applications that are running at a remote location, or a HTML-based web application. Multiple applications may appear to be contained within a single application container, or may live separately in multiple containers to simulate the appearance of multiple native applications. Usually only temporary cached data and/or a credential is persistent on the device itself.

Category 2: These are added to an application binary after compilation and dynamically modify the behaviour of the running application (for example to run the application within another sandbox and intercept and modify platform API calls) in an attempt to enforce data protection.

Category 3: The source code to the surrogate application is modified to incorporate a Software Development Kit (SDK) provided by the technology vendor. This SDK modifies the behaviour of standard API calls to instead call the SDK's API. The developer of the surrogate application will normally need to be involved in the wrapping process.

2.1.4. Application Wrappers 4.2 Security Requirements (iOS)

Category 1 technologies are essentially normal platform applications but which store and process minimal information, rather than deferring processing and storage to a central location. The development requirements for these applications are identical to other native platform applications. Developers should follow the guidelines given above.

Category 2 and 3 wrapping technologies are frequently used to provide enterprise management to applications via the MDM that the device is managed by. SDKs are integrated into these MDM solutions and can be used to configure settings in the application or to modify the behaviour of the application. For example, the application could be modified to always encrypt data or not use certain API calls.

On iOS, both category 2 and category 3 wrapping technologies require the surrogate developer's co-operation to wrap the application into a signed package for deployment onto an iOS device. As such, normally only custom-developed in-house applications, and sometimes B2B applications (with co-operation) can use these technologies.

As the robustness of these wrapping technologies cannot be asserted in the general case, they should not be used with an untrusted application; they should only be used to modify the behaviour of trusted applications, or for ease of management of the wrapped application. Preferably, in-house applications should be developed specifically against the previously described security recommendations wherever possible. The use of app-wrapping technologies should be seen as a less favourable alternative method of meeting the above security recommendations, where natively meeting them is not possible.

Ultimately, it is more challenging to gain confidence in an application whose behaviour has been modified by a category 2 technology. It is difficult to assert that dynamic application wrapping can cover all the possible ways an application may attempt to store, access and modify data. It is also difficult to make any general assertions about how any given wrapped application will behave. As such, the NCSC cannot give any assurances about category 2 technologies or wrapped applications in general, and hence cannot recommend their use as a security barrier at this time.

However, category 3 technologies are essentially an SDK or library which developers use as they would any other library or SDK. In the same way that the NCSC does not assure any standalone cryptographic libraries, we do not provide assurance in SDKs which wrap applications. The developer using the SDK should be confident of its functionality, as they would be with any other library.

3. GOOGLE

3.1. Core App Quality

Link: https://developer.android.com/docs/quality-guidelines/core-app-quality

3.1.1. VX-S1

> Notifications follow Material Design guidelines. In particular:
>
> Notifications are not used for cross-promotion or advertising another product, as this is strictly prohibited by the Play Store.
> Notification channels are defined according to best practices, rather than serving all notifications from one channel.
> Selecting the correct notification priority.
> Multiple notifications are stacked into a single notification group, where possible.
> Set timeouts for notifications where appropriate.
> Notifications are persistent only if related to ongoing events, such as music playback or a phone call.

3.1.2. PS-T5

> The app does not use non-SDK interfaces.

4. Open Web Application Security Project (OWASP)

4.1. Mobile Application Security Verification Standard (MASVS)

Link: https://github.com/OWASP/owasp-masvs/releases/tag/v1.4.2

4.1.1. 1.1 MSTG-ARCH-1

> All app components are identified and known to be needed.

4.1.2. 1.3 MSTG-ARCH-3

> A high-level architecture for the mobile app and all connected remote services has been defined and security has been addressed in that architecture.

4.1.3. 1.4 MSTG-ARCH-4

> Data considered sensitive in the context of the mobile app is clearly identified.

4.1.4. 1.5 MSTG-ARCH-5

All app components are defined in terms of the business functions and/or security functions they provide.

4.1.5. 1.7 MSTG-ARCH-7

All security controls have a centralized implementation.

4.1.6. 1.8 MSTG-ARCH-8

There is an explicit policy for how cryptographic keys (if any) are managed, and the lifecycle of cryptographic keys is enforced. Ideally, follow a key management standard such as NIST SP 800-57.

4.1.7. 1.10 MSTG-ARCH-10

Security is addressed within all parts of the software development lifecycle.

4.2. Application Security Verification Standard 4.0.3 (ASVS)

Link: https://raw.githubusercontent.com/OWASP/ASVS/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf

4.2.1. V1.1 Secure Software Development Lifecycle

1.1.2 Verify the use of threat modeling for every design change or sprint planning to identify threats, plan for countermeasures, facilitate appropriate risk responses, and guide security testing.

1.1.4 Verify documentation and justification of all the application's trust boundaries, components, and significant data flows.

1.1.5 Verify definition and security analysis of the application's high-level architecture and all connected remote services.

4.2.2. V1.2 Authentication Architecture

1.2.1 Verify the use of unique or special low-privilege operating system accounts for all application components, services, and servers.

4.2.3. V1.7 Errors, Logging and Auditing Architecture

V1.7 Errors, Logging and Auditing Architecture

4.2.4. V1.8 Data Protection and Privacy Architecture

1.8.1 Verify that all sensitive data is identified and classified into protection levels.

1.8.2 Verify that all protection levels have an associated set of protection requirements, such as encryption requirements, integrity requirements, retention, privacy and other confidentiality requirements, and that these are applied in the architecture.

4.2.5. V8.3 Sensitive Private Data

8.3.4 Verify that all sensitive data created and processed by the application has been identified, and ensure that a policy is in place on how to deal with sensitive data.