Windows Phone Platform Security

This article describes the Windows Phone (7) security model and compares and contrasts it with the security model used in Symbian. It is intended primarily for Symbian developers who are migrating to Windows Phone, but should also be useful for developers new to Nokia platforms, who want to understand the security model on both systems.

Developers that wish to understand the two models independent of each other can jump ahead to the Useful Links at the end of this article.

25 Jul 2011

Overview

The Windows Phone and Symbian security models are conceptually almost identical:

- They are designed to solve the same problems!
- They are based on a layered or tiered "least privilege" trust model
- They provide a capability based model for 3rd party applications to request access to protected functionality
- They run applications in an sandbox, isolating applications from each other
- They provide a protected data/file area that only the application can access
- Both use a testing and certification programme to determine which articles can be trusted with the capabilities

Developers who understand the Symbian Platform Security will have no difficulty understanding the Windows Phone security model, and visa versa. Indeed most developers will find the Windows phone security slightly simpler because it is easier to work out exactly which capabilities are required by your application.

The specific details of the security model tiers, how capabilities are managed, how the protected files are accessed and the signing program are of course different. These are each covered in the following sections.

Why platform security?

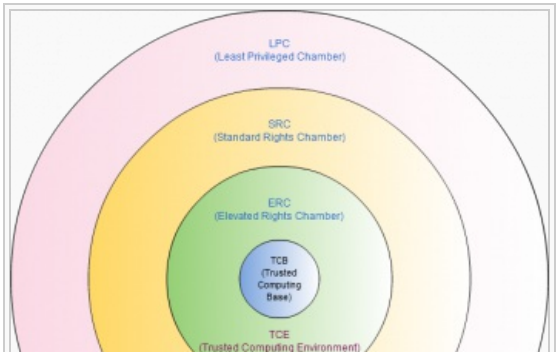
Many smartphone owners see their phone as their one essential communication tool, using it to keep in contact with peers, social networks and colleagues. As such, they expect their phones to be extremely reliable and to protect their private information, including contacts, emails, agenda, photographs, and location (to name a few), and they assume that the phone won't do anything to silently compromise their information or result in an unexpectedly large phone bill. The mobile device has to work all the time, even after installing new applications!

At the same time, even though the mobile device is more personal and often considered more essential than their desktop computer, mobile users don't want to be bothered by intrusive security, or to have to become computing experts in order to use their device safely. They certainly don't want to have to wait to make a phone call while their mobile device is running an anti-virus scanner!

These requirements can only be provided by a system wide security model that unobtrusively manages what applications can be installed, and what they can do on the phone.

Tiered layers of trust

Symbian and Windows phone both use a four-tiered security model to control access of code to device and platform services (windows phone refers to these tiers as "chambers"). The tiers/chambers for both platforms are shown in Figure 1 (Windows Phone naming shown above and Symbian below). Using a tiered system ensures that threats to the outer levels cannot be escalated as attacks on the inner levels - and that the inner levels are smaller and expose much less opportunity for attack in the first place.



The heart of both models is called the *Trusted Computing Base* (TCB). This tier contains just those services required to maintain the security model and its policies: the kernel and driver software, and the services that control software installation and file access.

The second layer(s) provides essential system services for use by other applications in a less privileged (but still very trusted) layer outside the TCB. Windows Phone calls this layer the *Elevated Rights Chamber* (ERC) while it is named the *Trusted Computing Environment* (TCE) in Symbian.

The third Windows Phone layer is the *Standard Rights Chamber* (SRC),

Figure 1: Tiers of Trust

which is used for *pre-installed* applications from Microsoft. The last layer is the *Least Privileged Chamber* (LPC) - this contains all non-Microsoft applications that are available through the Marketplace Hub.

The third Symbian layer is for *Signed* applications, and includes both both pre-installed and after market apps that have been signed with a digital certificate to indicate that they are trusted. In Symbian there is no differentiation in the layers for after market applications, however the actual testing required for pre-installed applications is usually more rigorous.

The last Symbian layer is *unsigned* (and includes "self-signed" applications), which indicates that the platform has no basis for trusting the application. Some less-sensitive operations may still be granted to these applications, but only following user approval.

The main difference between the models is that for the inner three chambers Windows phone uses security policies to define what code can do; capabilities are only used for after market applications in the LPC. In contrast Symbian uses capabilities for all layers - it is the presence of the capability that defines where the executable sits in the tiered model. In many cases this distinction is irrelevant since in either case you will need the direct support of OEM/Manufacturer to get access to the more sensitive functionality.

There are also some differences with respect to what untrusted code can do on each platform, and how they support beta testing and models like internal corporate application development. However Microsoft is gradually restricting its policies to address these cases.

Capabilities

Capabilities are access tokens that an application needs in order to be able to call certain functionality in code or access particular services. If you try to use a feature for which you don't have the capability you'll get an `UnauthorizedAccessException` when you first call that feature on Windows Phone or a `KErrPermissionDenied` (-46) Panic on Symbian.

Capabilities are requested at build time but only granted at install time to trusted applications. Usually "trusted" means that the application has been signed with a digital certificate that indicates that it has been through a testing programme. In the case of Symbian, this trust may also be granted by acknowledgement of a user prompt provided the application uses only a small set of less sensitive capabilities. Both Symbian and Windows Phone use capabilities to ensure that applications disclose what their code can do to end-users at install time or when viewing an application to purchase on Windows Phone Marketplace/Ovi.

The capability sets used by the two platforms are different. Symbian's capability set provides access to all system services, and individual capabilities therefore tend to provide broader access to a number of related system services. Windows phone capabilities comprise a smaller set of capabilities, covering the functionality most needed by *application developers* only. They tend to be more specific to a particular function or service.

The following table shows the Windows Phone capabilities, the "equivalent" Symbian capability, and a description/comparison. Note that the table does not display many of Symbian's system and manufacturer capabilities because there is no equivalent system access in Windows Phone.

WP	Symbian	Description/comparison
ID_CAP_APPOINTMENTS	ReadUserData/WriteUserData	Applications that access appointment/calendar data.
ID_CAP_CAMERA	MultimediaDD	Applications that use the camera capabilities. This value is for use by Mobile Operators and Original Equipment Manufacturers only. Application developers use ID CAP ISV CAMERA instead. This is equivalent to Symbian's `MultimediaDD` - you don't need it to access the Camera, but you do need it for "preferential" access over other applications.
ID_CAP_CONTACTS	ReadUserData/WriteUserData	Applications that access contact data.
ID_CAP_GAMERSERVICES	Not applicable	Applications that can interact with Xbox LIVE APIs. This must be disclosed due to privacy issues since data is shared with Xbox.

ID_CAP_IDENTITY_DEVICE	ReadDeviceData/WriteDeviceData	Applications that use device-specific information such as a unique device ID, manufacturer name, or model name.
ID_CAP_IDENTITY_USER	Not applicable	Applications that use the anonymous LiveID to uniquely identify the user in an anonymous fashion.
ID_CAP_ISV_CAMERA	UserEnvironment	Applications that use camera capabilities.
ID_CAP_LOCATION	Location	Applications with access to location services.
ID_CAP_MEDIALIB	NA	Applications that can access the media library. Symbian applications can access media on the device using the multimedia apis. Protected content may require the DRM capability.
ID_CAP_MICROPHONE	UserEnvironment	Applications that use the microphone. The application can record without a visual indication that recording is taking place.
ID_CAP_NETWORKING	NetworkServices	Applications with access to network services. This must be disclosed because services can incur charges when a phone is roaming.
ID_CAP_PHONEDIALER	NetworkServices	Applications that can place phone calls. This may happen without a visual indication for the end user.
ID_CAP_PUSH_NOTIFICATION	NetworkServices	Applications that can receive push notifications from an Internet service. This must be disclosed because usage could incur roaming charges. Symbian considers push notifications as "just another" Network service from a security perspective.
ID_CAP_SENSORS	ReadDeviceData/WriteDeviceData	Applications that use the Windows Phone sensors.
ID_CAP_WEBBROWSERCOMPONENT	NetworkServices	Applications that use the web browser component. There are security risks with scripting. Symbian treats web browser access as another network service

Symbian capabilities are defined in the project file (`.MMP` for a Symbian C++ application or `.PRO` for a Qt application). In general it is fairly straightforward to determine what capabilities are needed from the functionality used - guidance is provided in Qt & Symbian Platform Security and Fundamentals of Symbian C++/Platform Security#Working Out Which Capabilities You Need.

Windows phone capabilities are defined in the Windows Phone Application Manifest File, and again, it is usually fairly easy to work out which capabilities are needed. In addition, Microsoft provide the Capability Detection Tool to help - see How to: Use the Capability Detection Tool for Windows Phone for detailed instructions on its use.

Sandbox

Both Symbian and Windows Phone applications run in a "Sandbox"; a separate area which has a defined set of capabilities (or access control permissions) and which cannot interfere directly with other applications. In both cases this is achieved by running apps in a separate operating system process.

The level of interaction allowed between applications is dependent on the platform. Symbian provides mechanisms to allow applications to collaboratively communicate with each other, including: message queues, client-server, publish and subcribe, sockets, etc. It also provides mechanisms to allow *highly trusted* applications to interact with other application processes.

Windows phone is much more restrictive

- WP7 allows communication between applications only through Microsoft's cloud services.
- In WP 7.1 ("Mango") networking services (sockets) become available to applications, which will allow deeper communication.

Data caging/protected application data

Symbian and Windows Phone applications both provide applications with private and protected mechanisms for storing their data from other applications:

- Windows Phone (.NET) calls this concept "Isolated Storage" and offers areas for storing files and key value pairs, and local database storage. More information is available in the Isolated Storage Overview for Windows Phone
- Symbian calls this concept "data caging". The platform provide a private directory for every application that cannot be accessed by other applications. Apps can safely store their data using SQL databases or any other format. There are also other storage methods for settings, including the central repository.

The main difference between the two platforms is that Windows Phone applications can *only* access their own private area of the file system, while on Symbian, applications can also access shared areas.


Tip: This implies that at time of writing it is not possible to create your own services - you can't for example write your own music file type and share this with other music player applications.

Application signing and deployment

Symbian and Windows Phone both use application testing and signing programs to verify that third party applications can be trusted to run on the device. The test's main aim is to ensure that applications are reliable, behave as a user would expect, and that the phone remains reliable after they have been installed.

Applications that are sold through the app stores (Ovi or Windows Phone Marketplace) put additional conditions on applications in order to ensure that they are "morally and legally acceptable" - for example that they don't include pornography, use other's trademarks or make offensive or defamatory remarks. Microsoft also requires that applications must use Microsoft where applicable and may not mention other application stores.

The tests and conditions are similar in both places. The main difference between the platforms is that Symbian provides more distribution options (freedom for the developer to explore their own distribution and marketing models) while Windows Phone benefits from a single distribution portal and hence less opportunity for end-user confusion.

Windows Phone

Distribution to Windows Phone is available *only* through the Windows Phone Marketplace. A wide range of distribution models are supported, including free, paid, freemium (trial) and ad-funded. A "snapshot" of the current distribution policies is given in New Policies for Next Gen Windows Phone Marketplace. Details of the requirements, policies, and the public distribution model are given in the article linked below:

- Windows Phone Marketplace
  - Windows Phone Marketplace FAQ

    covers Registration, Application Submission, Certification, Pricing, Payout, Application Features, Promotional Opportunities, Regional Marketplace Information, Windows Phone Developer Support
  - Application Certification Requirements for Windows Phone

    covers Application Policies, Content Policies, Application Submission Requirements, Technical Certification Requirements, Additional Requirements for Specific Application Types
- Publishing Applications to the Windows Phone Marketplace

  covers Registration as a Windows Phone Developer, Developing and Test Your Application, Assembling the Prerequisites for Certification, Submitting Your Application for Certification, Link to Your Application in the Windows Phone Marketplace Catalog, Updating Your Application in the Windows Phone Marketplace, Support

In addition to the public distribution model, Microsoft is planning to make available two more distribution models in WP7.1 timeframes. The first is a Beta distribution model which allows you to distribute your application to up to 100 named users through the marketplace. The second is a "Private" marketplace, which is intended primarily to allow companies to distribute vertical apps to their users merely by emailing them a deep link to marketplace. Details on these programs are unavailable (at time of writing), however there are some interesting links below:

- Private Markets help make Windows Phone fit for business
- Microsoft readies private marketplace feature for Windows Phones

Symbian

Symbian applications that have been through the Symbian Signed certification and testing programme can be delivered through any channel, including your own website. Indeed applications that use only capabilities that a user can reasonably understand can be self-signed by the developer.

Developers that wish to benefit from Nokia's wide market reach and have access to features like operator billing can use Nokia's Nokia Store - this also provides options for free certification of applications sold through the store (only). Nokia requires that applications sold through the store comply with additional conditions which reduce the risk of legal action or harm to Nokia's reputation.

Links to the key Symbian signing and distribution documents are provided below: **Symbian Signed:**

- User guide: Symbian Signed
- User guide: Symbian Signed
- Symbian Signed Test Criteria
- Category:Symbian Signed
- www.symbiansigned.com

Nokia Store:

- Distributing your application
  - Packaging and Signing
  - Ovi Content Guidelines
  - Ovi Publish Checklist

In addition, Symbian provides various options for testing and beta testing, which are discussed briefly below.

Application testing during development

The platform security model(s) prevents unsigned applications from harming the phone - which creates a problem if you want to install your application during development or beta testing.

The solution on Windows Phone is that it will allow you to deploy up to 10 applications to a registered phone during testing (see Publishing Applications to the Windows Phone Marketplace, Deploying and Testing on a Physical Windows Phone Device and Windows Phone Developer Tools). For beta testing, the beta testing distribution model discussed in the preceding section should be sufficient for most developers. In addition, there is (unconfirmed by me) talk of a mechanism being provided to allow unlocking of developer phones - see ChevronWP7 Labs and ChevronWP7 Blogs for more information.

Symbian provides a number of mechanisms for allowing applications to be installed on devices during development. The best approach is to request a Developer Certificate which allows you to sign applications so that they will run on your specified set of devices. It is also possible to self-sign applications that only use basic capabilities.

Frequently Asked Questions

What tools are available to help me with security

Microsoft's software development life-cycle defines a number of security activities, and provides tools that support these. See Security for Windows Phone#Microsoft Security Development Lifecycle (SDL) for more information.

What is the security story for native code?

Windows Phone 7.1 supports only managed code (C#); native code (C/C++) or any approach that compiles to native code is not allowed for 3rd party application developers. Windows Phone 8.0 added native code capabilities for 3rd party developers. The same limitations as for managed code apply here as well.

Useful Links

Windows Phone

- Security for Windows Phone (MSDN)
- Windows Phone 7 Security and Management_FINAL_122010.pdf (White Paper, covers Windows Phone 7.0)
- Isolated Storage Overview for Windows Phone
- How to: Use the Capability Detection Tool for Windows Phone (MSDN)

- Application Manifest File for Windows Phone#Capabilities
- Blogs
  - Security for Windows Phone (www.syfuhs.net)
  - Windows phone capabilities security model (Jaime Rodriguez - blogs.msdn.com)

Qt

- Qt & Symbian Platform Security

Symbian C++

- Fundamentals of Symbian C++/Platform Security
- Symbian OS Platform Security (Book)
  - 01. Why a Secure Platform?
  - 02. Platform Security Concepts

Summary

Symbian and Windows Phone share very similar security models and testing programmes. Symbian offers slightly less restrictive development opportunities for developers in terms of sharing files and data between processes, but overall most developers will find that the security models are easy to understand and share more similarity than differences.