Shub-Nigurrath [ARTEAM]

Primer on Reversing Symbian S60 Applications

Version 1.4
Last Rev.: June 2007

Into this Tutorial

Author: Shub-Nigurrath

Forewords

Time has come to write a new tutorial. This time I will leave the Win32/Windows world to open a new window (☺) into another world.

Symbian phones, mostly Nokia, have started to become really interesting and powerful. Important applications appeared on the marked and the system became a really general purpose operating system. The applications started to be protected with important tricks and some tools appeared which can handle them.

Unfortunately the Symbian scene is not so prolific of tutorials and what I found after a lot of searching and talking with others guys are just a few simple and quite old tutorials and few advanced things, mostly not written in English.

I decided then to take a long journey into this world, examining which tools you can use to disassemble the Symbian programs, how to approach to them and what generally you can do to create and distribute patches for those applications.

I started from the ground up, just because as said there were no discussion forum like our (at least I have not found them) where one can ask, the present special issue collects a series of single tutorials I wrote with different targets and difficulty levels. Probably the few Symbian groups around will laugh at me for the simple or even not correct approach, but as usual if one knows things better he should write a tutorial to demonstrate it.

The tutorial will cover different issues:

- Few words on the Symbian OS
- What instruments we have and what to use and customize them (particularly IDA)
- Practical examples of real applications

I also included a long list of references and further readings, as usual.

The commercial applications I used have been selected using two criteria:

- being educative for my purposes
- being already cracked by someone else, so as to not create problems on my own

Have phun,
Shub

Disclaimers

All code included with this tutorial is free to use and modify; we only ask that you mention where you found it. This tutorial is also free to distribute in its current unaltered form, with all the included supplements.

All the commercial programs used within this document have been used only for the purpose of demonstrating the theories and methods described. No distribution of patched applications has been done under any media or host. The applications used were most of the times already been patched, and cracked versions were available since a lot of time. ARTeam or the authors of the paper cannot be considered responsible damages the companies holding rights on those programs. The scope of this tutorial as well as any other ARTeam tutorial is of sharing knowledge and teaching how to patch applications, how to bypass protections and generally speaking how to improve the RCE art. We are not releasing any cracked application.

Verification

ARTeam.esfv can be opened in the ARTeamESFVChecker to verify all files have been released by ARTeam and are unaltered. The ARTeamESFVChecker can be obtained in the release section of the ARTeam site: http://releases.accessroot.com

Table of Contents

1. Few words on what Symbian S60 is

I specifically concentrated on the Symbian OS v7.0 S60 operating system, almost because I have a Nokia 6600 model. Usually this version is called Symbian S60 (not to be confused with S60 3rd edition).

Unfortunately, by the reversing point of view, there are several versions of the Symbian OS and several series, most of which are not compatible at binary level, but all shares a common set of functionalities. Programs are distributed by producers for each phone model, usually you have to select the phone model and then the distributor's site will give you the correct version of the program.
This is IMHO one of the reasons why there are not much tutorials around on this OS. After all these phones are working with ARM processors which at assembler levels are all the same thing.
The fact that ARM is a RISC processor (Reduced Instruction Set CPU, so more difficult assembler) is IMHO another important obstacle.

Symbian is currently owned by Ericsson (15.6%), Nokia (47.9%), Panasonic (10.5%), Samsung (4.5%), Siemens AG (8.4%), and Sony Ericsson (13.1%).

Consider that even for just the Symbian versions 7.0 there are different Series to consider, which differs mainly for the users interfaces and services:

- S60, released in 2002, it's one of the most used versions, used for a lot of top-selling phones (Nokia 6600, 6630, N-gage and all the N series, like recent N-70) and absolutely the most popular and stable version.
- S80, used mostly by communicators (like Nokia 9210)
- S90, another version less spread especially developed for supporting special devices. Unfortunately, Series 90 is completely incompatible with Series 60.

The main CPUs of all these phones are ARM compatible chips.

Here I report using the Wikipedia page (http://en.wikipedia.org/wiki/Symbian_OS) the most relevant today Symbian releases.

- Symbian OS v7.0 and v7.0s. First shipped in 2003. This is an important Symbian release which appeared with all contemporary user interfaces including UIQ (Sony Ericsson P800, P900, P910, Motorola A925, A1000), Series 80 (Nokia 9300, 9500), Series 90 (Nokia 7710), Series 60 (Nokia 6600, 7310) as well as several FOMA phones in Japan.
  In 2004, Psion sold its stake in Symbian.
  Also in 2004, the first worm for mobile phones using Symbian OS, Cabir, was developed, which used Bluetooth to spread itself to nearby phones.

- Symbian OS v8.0. First shipped in 2004, one of its advantages would have been a choice of two different kernels (EKA1 or EKA2). However, the EKA2 kernel version did not ship until SymbianOS v8.1b. The kernels behave more or less identically from user-side, but are internally very different. EKA1 was chosen by some manufacturers to maintain compatibility with old device drivers, whilst EKA2 offered advantages such as a hard real-time capability. v8.0b was deproductized in 2003.

- Symbian OS v8.1. Basically a cleaned-up version of 8.0, this was available in 8.1a and 8.1b versions, with EKA1 and EKA2 kernels respectively. The 8.1b version, with EKA2's single-chip phone support but no additional security layer, was popular among Japanese phone companies desiring the real-time support but not allowing open application installation.

- Symbian OS v9.0. This version was used for internal Symbian purposes only. It was deproductised in 2004. v9.0 marked the end of the road for EKA1. v8.1a is the final EKA1 version of SymbianOS.
  Symbian OS has generally maintained reasonable binary compatibility. In theory the OS was BC from ER1-ER5, then from 6.0 to 8.1b. Substantial changes were needed for 9.0, related to tools and security, but this should be a one-off event. The move from requiring ARMv4 to requiring ARMv5 did not break backwards compatibility.
  Anyway Symbian developer proclaims that porting from Symbian 8.x to Symbian 9.x is a more daunting process than Symbian says.

- Symbian OS v9.1 released early 2005. It includes many new security related features, particularly a controversial platform security module facilitating mandatory code signing. Symbian argues that applications and content, and therefore a developers investment, are better protected than ever, however others contend that the requirement that every application be signed (and thus approved) violates the rights of the end-user, the owner of the phone, and limits the amount of free software available. The new ARM EABI binary model means developers need to retool and the security changes mean they may have to recode. S60 3rd Edition phones have Symbian OS 9.1. Sony Ericsson is shipping the M600i based on Symbian OS 9.1 and should ship the P990 in Q3 2006. The earlier versions had a fatal defect where the phone hangs temporarily after the owner sent hundreds of SMSes :-O

- Symbian OS v9.2 released Q1 2006. Support for Bluetooth 2.0 (was 1.2) and OMA Device Management 1.2 (was 1.1.2). S60 3rd Edition Feature Pack 1 phones have Symbian OS 9.2.

- Symbian OS v9.3 released on 12 July 2006. Upgrades include native support for WiFi 802.11, HSDPA, Vietnamese language support.

On November 16, 2006, the 100 millionth smartphone running the OS was shipped.

1.1. Few details on the operative system

Symbian OS is an operating system, designed for mobile devices, with associated libraries, user interface frameworks and reference implementations of common tools, produced by Symbian Ltd. It is a descendant of Psion's EPOC and **runs exclusively on ARM processors**.

There are multiple platforms, based upon Symbian OS, which provide an SDK for application developers wishing to target a Symbian OS device - the main ones being UIQ, Series 60, etc. Individual phone products, or families, often have SDKs or SDK extensions downloadable from the manufacturer's website too. The SDKs contain documentation, the header files and library files required to build Symbian OS software, and a Windows-based emulator ("WINS"). Up until Symbian OS version 8, the SDKs also included a version of the GCC compiler (a cross-compiler) required to build software to work on the device.

Unfortunately WINS is an Emulator and not a Simulator. The difference is great: emulators emulate the real device but are indeed Win32 or PC programs which are built to look like real phones, simulators instead, using real device ROMs acts just like the real devices doing calls to the ROM of the device. This difference, as we will see doesn't allow simulating **any** phone application on a PC. You can only execute on the PC just those applications for which you have the sources and that you are able to compile: you must compile the application telling that the destination platform is WINS. This is a limitation by design and to do a real on-device debugging you can buy a tool from CodeWarrior[1] or buy a hardware card which allows remote debugging on the phone from the PC.

Symbian OS 9 requires a new compiler - a choice of compilers is available including a new version of GCC (the free C/C++ compiler of Linux, available also on Windows). In terms of SDKs, UIQ Technology now provides a simplified framework so that the single UIQ SDK forms the basis for developing on all UIQ 3 devices, such as the Sony Ericsson P990 and Sony Ericsson M600.

Symbian C++ programming is commonly done with a commercial IDE. For current versions of Symbian OS, CodeWarrior for Symbian OS is favored. The CodeWarrior tools will be replaced during 2006 by Carbide.c++, an Eclipse-based IDE developed by Nokia. It is expected that Carbide.c++ will be offered in different versions: a free version may allow users to prototype software on the emulator for the first time in a free product.

Visual Basic, VB.NET, and C# development for Symbian can be accomplished through AppForge Crossfire, a plugin for Microsoft Visual Studio (Appforge will be discussed into another specific tutorial).

There's also a version of a Borland IDE for Symbian OS. Symbian OS development is also possible on Linux and Mac OS X using tools and techniques developed by the community, partly enabled by Symbian releasing the source code for key tools.

Once developed, Symbian OS applications need to find a route to customers' mobile phones. They are packaged in SIS files which may be installed over-the-air, via PC connect or in some cases via Bluetooth or memory cards. An alternative is to partner with a phone manufacturer to have the software included on the

[1] http://seap.forum.nokia.com/info/sw.nokia.com/id/204ab18e-410c-4c59-bcd4-dda936c8a79b/CodeWarrior_On_Device_Debug_Kit_for_Series_60_3rd_Edition.html

phone itself. The SIS file route will be a little more difficult from Symbian OS 9, because any application wishing to have any capabilities beyond the bare minimum must be signed via the "Symbian Signed" program.

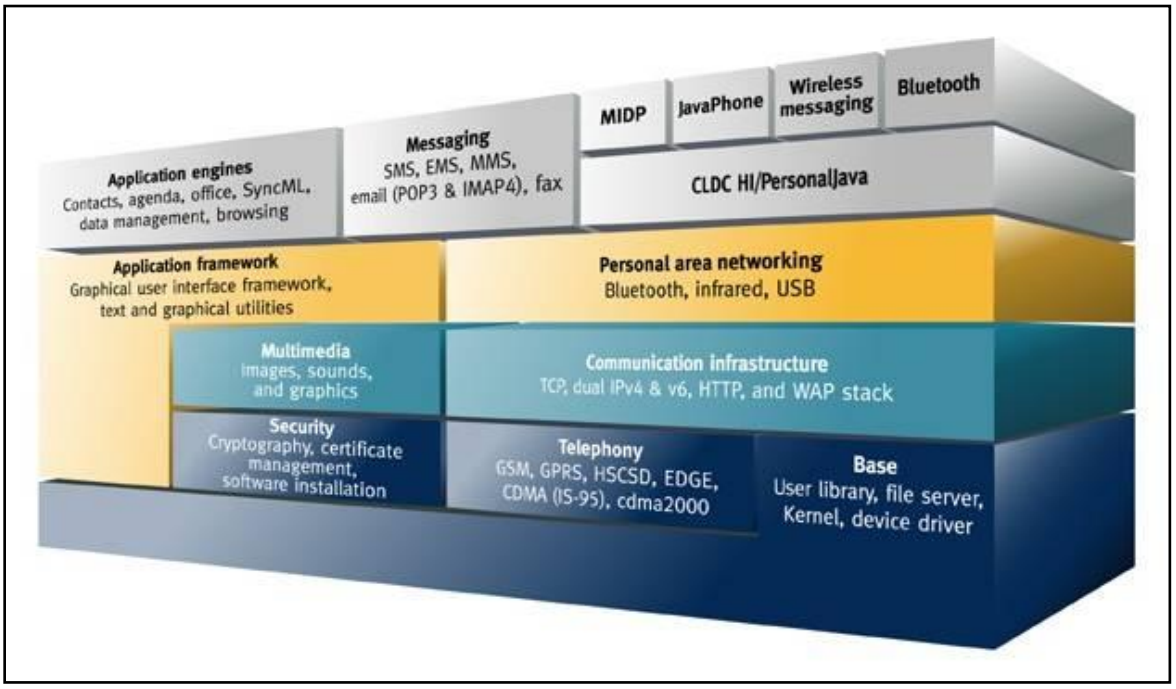

Figure 1 – Overview of SymbianOS Services

As shown in Figure 1 it's not only possible to develop using C/C++ or VB bridging frameworks, but also using Java. Java ME applications for Symbian OS are developed using standard techniques and tools such as the Sun Java Wireless Toolkit (formerly the J2ME Wireless Toolkit). They are packaged as JAR (and possibly JAD) files. Both CLDC and CDC applications can be created with NetBeans. SymbianOS also support the MIDP 2.0 Java specifications (a particular standard java profile/framework for mobile phones, common to most modern models), beside the java support. These specifications are meant to be vendor and model independent. Reversing and patching Java mobile applications is the simplest thing.

Nokia Series 60 phones can also run python scripts when the interpreter is installed, with a custom made API that allows for Bluetooth support and such. There is also an interactive console to allow the user to write python scripts directly from the phone.

1.1.1 Phone Drives structure

Few short things might become useful later on and are required to find data around [9].

Symbian file system is based on drive letters, directories and files

- C: FLASH RAM User data and user installed applications
- D: TEMP RAM Temporary file storage for applications
- E: MMC card Removable disk for pictures and applications
- Z: OS ROM Flash drive that contains most of the OS files

All drives have System directory:

- The directory is created automatically on a new media when one is inserted
- The System directory contains directory tree that contains OS and application files. Very much the same as C:\windows

Most important directories:

- System\Apps Applications that are visible to user
- System\Recogs Recognizer components
- System\Install Data needed for uninstallation of user installed applications
- System\libs System and third party libraries

Implementation Of User Services.