How to troubleshoot Platform Security issues

During application development one usualy faces various problems related to the Platform Security concepts. It is a little bit tricky to find out whether a problem we are facing is related to Platform Security or not. The check list below will help analyzing the problem related to Platform Security.

Check the emulator log output

The first thing that is easy to check is the emulator log output. The emulator log output can be enabled from the Tools | Preferences menu. Make sure that "Enable EPOCWIND.OUT logging" is checked. Alternatively, the log can be enabled from the `epoc.ini`. Make sure that `LogToFile` has the value of 1.

```
LogToFile 1
```

Run the application, do something and then exit. Then, open `EPOCWIND.OUT` file, which is located at %TEMP% folder. Analyze the file whether there is `PlatSec` error message. The following example shows an example where the application has a problem related to Platform Security.

```
151.975 *PlatSec* WARNING - Capability check would have failed - A Message
(function number=0x00000000) from Thread Screenshot[20000555]0001::screenshot,
sent to Server !Windowserver, was checked by Thread
WSERV.EXE[10003b20]0001::Wserv and was found to be missing the capabilities:
SwEvent .  Additional diagnostic message: Capability check failed for
RWindowGroup::CaptureKeyUpsAndDowns API

   153.600 *PlatSec* WARNING - Capability check would have failed - A
Message (function number=0x00000000) from Thread
Screenshot[20000555]0002::screenshot, sent to Server !MsvServer, was checked by
Thread !MsvServer[1000484b]0001::!MsvServer and was found to be missing the
capability: ReadUserData.
```

The example shows that the application is missing two capabilities, i.e. **SwEvent** and **ReadUserData**.

Perform capability checks on emulator

This is basically similar to the previous one. Instead of writing the problem to a log file, emulator will notify immediately when there is a problem related to Platform Security.

Emulator capability checks can be enabled from the Tools | Preferences menu on the emulator. It can be enabled from `epoc.ini` file as well. Change the value `PlatSecDisabledCaps` to `NONE`.

```
PlatSecDisabledCaps NONE
```

It means that there is no disabled capability. In other words, all capabilities check are enabled.

When this feature is enabled, emulator will display a dialog each time there is call to restricted API. The picture below shows an example of error message that may come up because of missing capabilities.

(C) Copyright Nokia 2014. All rights reserved.

What APIs am I using and what capabilities do I need?

After a problem is detected, the next step is how to find where the problem comes from. Debugging will always help. Besides that, check also the SDK Help to see whether the application is any missing capabilities. The following example shows an example of how the capabilities information written in the SDK Help.

```
Capability Information
Required capabilities:
ReadUserData WriteUserData ReadDeviceData WriteDeviceData
```

Check the MMP file

After all the list of capabilities have been gathered, the next step is to check the `.mmp`. Make sure that all the capabilities are listed there.

```
CAPABILITY SwEvent LocalServices NetworkServices ReadUserData WriteUserData
```

Note that Carbide.c++ 1.1 does not use `.mmp file`. To modify capabilities on Carbide.c++ 1.1, right click the project and select Properties | C/C++ Build.

Carbide.c++ Capability Scanner

There is a new plug-in for Carbide.c++, called Capability Scanner], that is able to detect Platform Security problem from the IDE. The plug-in is available starting with the Carbide.c++ release v1.3.

Check DLL dependencies

In Platform Security, an executable can only load DLL that has more capabilities than its own. For example, an executable with capabilities of A+B cannot load a DLL with capabilities of A only. Make sure that all DLLs that are loaded by the executable have more capabilities.

This problem normally arises when using third party DLLs. Most DLLs on the ROM have "All -Tcb"; so most third party applications shall have less capabilities than DLLs on the ROM.

Checking the capabilities of an executable, including `.exe` and `.dll` can be done using elftran tool. Run the tool using "-dump h" parameter and check capabilities section.

```
ELFTRAN - ELF file preprocessor V02.01 (Build 564)
```


(C) Copyright Nokia 2014. All rights reserved.

```
Copyright (c) 1996-2006 Symbian Software Ltd.

E32ImageFile 'screenshot.exe'
Entry points are not called
...
Capabilities: 00000000 0001f000
...
```

The executable of the example above has the capabilities of 0x0001 F000. The definition of the capabilities bits can be found in `e32capabilities.h` header file. It is shipped with any Symbian OS 9 SDKs, including Symbian 3rd Edition SDK.

There is no such file as e32capabilities.h in either MR release or the FP1 release, there is a file named e32capability.h but that is not usefull at all.

However it is probably easier to use *elftran -dump s <filename>* to get the capabilities of an executable file.

```
ELFTRAN - ELF file preprocessor V02.01 (Build 573)
Copyright (c) 1996-2006 Symbian Software Ltd.

E32ImageFile 'HelloWorldBasic.exe'
Secure ID: 102ee322
Vendor ID: 10333333
Capabilities: 00000000 000ffffe
              CommDD
              PowerMgmt
              MultimediaDD
              ReadDeviceData
              WriteDeviceData
              DRM
              TrustedUI
              ProtServ
              DiskAdmin
              NetworkControl
              AllFiles
              SwEvent
              NetworkServices
              LocalServices
              ReadUserData
              WriteUserData
              Location
              SurroundingsDD
              UserEnvironment
```

Internal links

- Platform Security
- E32Image file format on Symbian OS 9
- Epocwind.out

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.