Status Update 2

SECRET//NOFORN

Status Update 2 – Last Updated July 12, 2013

Objective

My goal was to better understand how the Siemens phone application uses the ifx_mps driver. The first step was to determine which processes were opening the ifx_mps device files. I built strace and lsof for the phone and put them in /usr/sbin. Using lsof, I determined that SvcConfig and its threads (total 70 of 95) are the only processes that open /dev/ifx_mps/cmd. In the current state, no ifx_mps channels are opened by any process. After closer examination, the 70 SvcConfig processes have the following command line:

```
SvcConfig services.conf -startLogDaemon -logAll V2 R0.92.0      HFA  120822


lsof | grep ifx_mps | wc -l

ps -ef | grep SvcConfig | wc -l
```

The SvcConfig (using PID 503 as an example) process opens the following files (in addition to numerous sockets and pipes filtered out of the result below):

```
SvcConfig   503       ???  cwd       ???                ???       ???         ??? /Op
era_Deploy

SvcConfig   503       ???  exe       ???                ???       ???         ??? /Op
era_Deploy/SvcConfig

SvcConfig   503       ???    0       ???                ???       ???         ??? /de
v/null

SvcConfig   503       ???    1       ???                ???       ???         ??? /de
v/null

SvcConfig   503       ???    2       ???                ???       ???         ??? /de
v/null

SvcConfig   503       ???   10       ???                ???       ???         ??? /Op
era_Deploy/healthservice.conf

SvcConfig   503       ???   24       ???                ???       ???         ??? /da
ta/database/phone.db
```

```
SvcConfig   503         ???   37        ???                 ???       ???         ??? /de
v/input/keyboards

SvcConfig   503         ???   38        ???                 ???       ???         ??? /de
v/input/keyInput

SvcConfig   503         ???   39        ???                 ???       ???         ??? /de
v/input/HookSw

SvcConfig   503         ???   40        ???                 ???       ???         ??? /de
v/sidecar

SvcConfig   503         ???   41        ???                 ???       ???         ??? /de
v/ledmatrix

SvcConfig   503         ???   42        ???                 ???       ???         ??? /de
v/fb/0

SvcConfig   503         ???   53        ???                 ???       ???         ??? /tm
p/lldpfifo

SvcConfig   503         ???   56        ???                 ???       ???         ??? /tm
p/LldpManagerFifo

SvcConfig   503         ???   62        ???                 ???       ???         ??? /de
v/pc_status

SvcConfig   503         ???   64        ???                 ???       ???         ??? /de
v/ifx_mps/cmd

SvcConfig   503         ???   81        ???                 ???       ???         ??? /Op
era_Deploy/Mobile_0100_base.dls

SvcConfig   503         ???  100        ???                 ???       ???         ??? /de
v/sidecar

SvcConfig   503         ???  mem        ???               1f:04         0         386 /Op
era_Deploy/SvcConfig

SvcConfig   503         ???  mem        ???               1f:04     20480         386 /Op
era_Deploy/SvcConfig
```

The next step was to begin decomposing SvcConfig and the services.conf file. SvcConfig – and the Opera executables in general – is a C++ application making heavy use of shared libraries and a distributed object framework with separate client-side proxy libraries and server-side invoker libraries although there is no machine or processor boundary between the client and server side code (i.e. they both run on the Linux OSand NOT the voice co-processor).

Future Approaches

Hook syscalls to ifx_mps

Intercept reads, writes, and ioctls to ifx_mps. This would require observing normal operation to determine the functionality we'd want to create at this level; and some of the functionality we desire is not observable because it is abnormal.

Reverse Opera proxy libraries

Begin reversing the Opera C++ proxy libraries (client) and write an application that uses them. Without header files, this would involve reversing to determine appropriate object formats and parameters for calling these functions.

Possible places to start:

```
  Library name                            Creator fn                              Sup
ported interface        Supported Protocol

#####################################################################################
##########################################

libPhysicalInterfaceService.so          createphysicalEventObserverProxy        Physi
calEventObserver           opera_text

libPhysicalInterfaceService.so          createfunctionalEventObserverProxy      Funct
ionalEventObserver         opera_text

libPhysicalInterfaceServiceProxy.so     createphysicalEventGenerationProxy      Physi
calEventGenerationIfc      opera_text

libToneGenerationServiceProxy.so        createToneGenerationServiceProxy        ToneG
enerationServiceIfc        opera_text

libToneGenerationService.so             createToneGenerationEventObserverProxy  ToneG
enerationEventObserverIfc  opera_text

libMediaControlServiceProxy.so          createMediaControlServiceProxy          Media
ControlServiceIfc          opera_text

libVoiceEngineProxy.so                  createVoiceEngineProxy                  Voice
EngineInterface            opera_text

libMediaControlServiceProxy.so          createAuditoryDeviceProxy               Audit
oryDeviceIfc               opera_text

libMediaControlService.so               createAcousticStateEventObserverProxy   Acous
ticStateEventObserver      opera_text

libCommunicationsServiceProxy.so        createCommunicationsServiceProxy        Commu
nicationsServiceIfc        CommunicationsServiceProtocol

libCommunicationsService.so             createCommunicationsServiceEventProxy   Commu
nicationsServiceEventIfc   CommunicationsServiceProtocol
```

Reverse Opera invoker libraries

Begin reversing the Opera C++ invoker libraries (server) to understand how they interface with the ifx_mps driver(s); as I currently believe the invoker libraries contain the code that receive high-level requests from the client-proxies and communicates with the ifx_mps driver.