WIDEVINE®

Widevine Level 3 OEMCrypto Guide

Revision History

Version	Date	Description	Author
1	10/13/2017	Initial revision.	Srujan Gaddam
2	12/14/2017	Updated implementation requirements and integration.	Srujan Gaddam
3	5/3/2018	Minor changes to provisioning information.	Srujan Gaddam
4	8/2/18	Refactor keybox to device keys.	Srujan Gaddam
5	4/1/19	Clarify requirements and integration.	Srujan Gaddam

Table of Contents

Background

The Level 3 OEMCrypto is a software-only implementation of OEMCrypto provided by Widevine. It is used in conjunction with the Content Decryption Module (CDM) to provide content protection for devices that do not have a Trusted Execution Environment (TEE). These devices are hence referred to as Level 3. Provisioning is done via a field-provisioned keybox for v13, and from OEMCrypto v14 onwards, an OEM Certificate and private key.

Refer to *Widevine Modular DRM Security Integration Guide for Common Encryption (CENC)* for in-depth information on OEMCrypto and *Widevine CE CDM Integration Guide* for information on how the CDM works and how to build it.

Audience

This short guide is meant for partners who require a Level 3 solution for their devices. They must work with content providers to determine if this solution is right for them.

Process

Since the Level 3 OEMCrypto software we provide is platform specific, we need to know what kind of devices you wish to install it on and their CPU architecture/endianness (e.g. arm, x86_64, mipsel64, etc.). Furthermore, as part of provisioning, each model will be given a system ID, so the keybox/certificate can be unique to the device model. Partners who are integrating with multiple device types will need separate system IDs. We then provide a build for each architecture including the CDM which the partner integrates into their application. It is up to the partner if they want a separate system ID per architecture across the same model, but we recommend keeping them separate in case a certain architecture becomes compromised.

Implementation Requirements

There are a few code implementations we require from partners that we use as part of our Level 3 OEMCrypto implementation.

One such function returns a unique device ID (named **getUniqueID**), so device keys can be encrypted and decrypted for a specific device only. This **must be consistent and unique** for that device, or device keys will not be decrypted properly, meaning playback can not properly occur. The interface for this is included as part of oemcrypto/include/level3.h and its implementation reference is included in oec_level3.gyp.

Another requirement is an implementation of a file system object for OEMCrypto to use in storing device keys and other data. We call this the **OEMCrypto_Level3FileSystem**. The interface exists under the header level3_file_system.h in the oemcrypto/include/ subfolder. It is up to the partner to provide a location for the Level 3 OEMCrypto to read and write files.

As part of this, partners are required to implement factory and delete/recycle methods **(createLevel3FileSystem** and **deleteLevel3FileSystem**) as well, which creates the object and deletes it when OEMCrypto initializes and terminates. The interface for this exists under oemcrypto/include/level3.h. Keep in mind this is separate from any file storage requirements the CDM needs. Note that we include a test version of the OEMCrypto_Level3FileSystem to use for the unit tests that **will not work** for production, since it does not actually write to the file system.

Lastly, we require partners to implement a function that generates 64-bit random seeds (named **generate_entropy**) *if necessary*. This is used to seed an internal RNG for encryption purposes. We provide an implementation already for this function under oemcrypto/level3/generate_entropy_linux.cpp, which we suggest partners use, **but only if /dev/urandom both exists and is readable on the device**. If this is not possible with the device, partners **must** implement this function. The interface for this also exists under oemcrypto/include/level3.h.

Integration

Inside of the CDM package we deliver, there are gyp build files needed for building the software. For Level 3 specifically, we include a file called oemcrypto/level3/oec_level3.gyp, which has references to the files (some of which we don’t ship, since they are part of implementation requirements) needed for building the Level 3 OEMCrypto. The bulk of the API implementation is embedded into one obfuscated monolithic file called libl3oemcrypto.cpp, which is built for the specific device and implements all the necessary OEMCrypto methods.

For CE partners, we supply a static OEMCrypto adapter to build with Level 3. To integrate the Level 3 with the CDM, you include the oec_level3_static dependency and the OEMCrypto_Level3FileSystem implementation and its factory methods. For an example, take a look at cdm_unittests.gyp and oemcrypto_unittests.gypi. Substituting ‘oemcrypto_lib%’ with ‘level3’ will build the tests with the Level 3 OEMCrypto.