Verified Boot

a38x, u-boot, uboot, bootloader

Verified Boot is a way to ensure that only authenticated code will be executed on a machine. This page provides instructions on setting this up for the startup phase from u-boot to Linux.

Configure U-Boot to support booting signed FIT images

For U-Boot verified boot requires using the FIT image format, however this is often not enabled by default. The followin configuration options have to be selected in u-boot's menuconfig:

```
CONFIG_FIT
CONFIG_FIT_SIGNATURE
CONFIG_RSA
```

It is important to keep the newly built tools/mkimage executable available for the next steps!

Create FIT image

FIT images contain all files required to boot a kernel, such as the kernel image, device-tree-blob and possibly a ramdisk. They are handled by the bootm command, as the only argument.

First a description file has to be created for the image. It follows the DeviceTree language and is fully described in *doc/uImage.FIT/source_file_format.txt*.

This is a simple example for the Clearfog board:

```
/*
 * Simple U-Boot uImage source file containing a single kernel and FDT blob
 */

/dts-v1/;

/ {
    description = "Simple image with single Linux kernel and FDT blob";
    #address-cells = <1>;

    images {
        kernel@1 {
            description = "Vanilla Linux kernel";
            data = /incbin/("/boot/zImage");
            type = "kernel";
            arch = "arm";
            os = "linux";
            compression = "none";
            load = <0x00008000>;
            entry = <0x00008000>;
```

```
            hash@1 {
                algo = "crc32";
            };
            hash@2 {
                algo = "sha1";
            };
        };
        fdt@1 {
            description = "Flattened Device Tree blob";
            data = /incbin/("/boot/dtb/armada-388-clearfog.dtb");
            type = "flat_dt";
            arch = "arm";
            compression = "none";
            hash@1 {
                algo = "crc32";
            };
            hash@2 {
                algo = "sha1";
            };
        };
    };

    configurations {
        default = "conf@1";
        conf@1 {
            description = "Boot Linux kernel with FDT blob";
            kernel = "kernel@1";
            fdt = "fdt@1";
        };
    };
};
```

From this file, and the referenced kernel Image and device-tree file, an FIT image can be built using u-boot's mkimage. It is very important to use the version that has been built in the previous step!

```
mkimage -f kernel_fdt.its fitImage
```

fitImage is the output file, and seems to be a standard name for these images. At least one target in mainline u-boot uses it (and there are few targets that enable FIT).

To try if it works, put the fitImage into /boot, and use bootm to execute it. Sample for Clearfog (adapt as required):

```
ext4load mmc 0:1 0x02000000 /boot/fitImage
bootm 0x02000000
```

If everything went well, the system should boot just as without the fit image.

Create Signing Key

This is quite easy and was copied from u-boot documentation:

```
mkdir keys
openssl genpkey -algorithm RSA -out keys/dev.key -pkeyopt
rsa_keygen_bits:2048 -pkeyopt rsa_keygen_pubexp:65537
openssl req -batch -new -x509 -key keys/dev.key -out keys/dev.crt
```

The next section expects the key to be in a folder called *keys*, but it might just as well be anywhere else!

Sign FIT image

For signing these images, the description file has to be extended by a signature sections. They belong under each section that is to be signed: kernel@1, kernel@2, fdt@1, conf@1, ...

Below is another sample for the Clearfog, this time with signature sections for kernel and DeviceTree:

```
/*
 * Simple U-Boot uImage source file containing a single kernel and FDT blob
 */

/dts-v1/;

/ {
    description = "Simple image with single Linux kernel and FDT blob";
    #address-cells = <1>;

    images {
        kernel@1 {
            description = "Vanilla Linux kernel";
            data = /incbin/("/boot/zImage");
            type = "kernel";
            arch = "arm";
            os = "linux";
            compression = "none";
            load = <0x00008000>;
            entry = <0x00008000>;
            hash@1 {
                algo = "crc32";
            };
            hash@2 {
                algo = "sha1";
            };
            signature@1 {
                algo = "sha1,rs2048";
            };
        };
```

```
        fdt@1 {
            description = "Flattened Device Tree blob";
            data = /incbin/("/boot/dtb/armada-388-clearfog.dtb");
            type = "flat_dt";
            arch = "arm";
            compression = "none";
            hash@1 {
                algo = "crc32";
            };
            hash@2 {
                algo = "sha1";
            };
            signature@1 {
                algo = "sha1,rs2048";
            };
        };
    };

    configurations {
        default = "conf@1";
        conf@1 {
            description = "Boot Linux kernel with FDT blob";
            kernel = "kernel@1";
            fdt = "fdt@1";
        };
    };
};
```

Now FIT images can still be built like in the previous section, but then they would NOT be signed. Instead, the location of the signing key needs to be passed to mkimage. Here it is assumed that all keys are in a folder called *keys*, in the current directory:

```
./tools/mkimage -f kernel_fdt_signed.its -k keys fitImage
```

Look for *Sign algo* and *Sign value* fields in the onscreen output. If there is no signature value, something went wrong!

It might make sense to sign the configuration, instead of kernel and dtb. This can prevent a mix-and-match attack in case there are multiple kernels / DTBs / ramdisks available in the image. According to the documentation, a signature for a configuration will also contain signatures for the hashes of all files that are part of the configuration. So signatures are then not required in kernel and fdt sections. FIT images can also be signed *after* creating them, by using mkimage's *-f* and *-F* options!

Include Public Signing Key in U-Boot

For signature validation to be of any use, the public key has to be available **before** loading the FIT image. This is why U-Boot expects it in the included DeviceTree file. It is most probably one of *arch/arm/dts/*.dts*, and compiled into both dts/dt.dtb, and u-boot.dtb. This can be done manually, but mkimage provides a much easier automatic method via its *-K* parameter:

```
./tools/mkimage -f kernel_fdt_signed.its -k keys -K dts/dt.dtb -r image
fitImage
```

The *-r* option tells u-boot which signatures are **required** to be checked at boot. When it is omitted, u-boot will happily boot any signed, unsigned or wrongly signed images. This option takes two possible values: *image* and *conf*. The latter should be used if the configuration section is signed!

Now the changed dtb has to be embedded into the U-Boot SPL. This can easily be achieved by rerunning make*Italic Text*, while triple-checking that dts/dt.dtb is **not** changed by make.

This section was only tested on the Clearfog. Especially the path dts/dt.dtb **might** be different for other targets.

Additional Resources

- FIT image description file format: http://git.denx.de/?p=u-boot.git;a=blob_plain;f=doc/uImage.FIT/source_file_format.txt;hb=HEAD
- Full documentation on u-boot signatures: http://git.denx.de/?p=u-boot.git;a=blob_plain;f=doc/uImage.FIT/signature.txt;hb=HEAD
- Sample for signing *conf* section: http://git.denx.de/?p=u-boot.git;a=blob_plain;f=doc/uImage.FIT/sign-configs.its;hb=HEAD