Signing firmware

Firmware signing is a process in which the firmware is given a digital signature. The purpose of this signature is currently limited to checking whether the firmware or reloadable parts of it (=AVM plug-ins) come from a trustworthy source and can therefore be flashed or reloaded and integrated. With regard to the firmware, this mechanism has only been activated by AVM since Fritz!OS-6.5x. Since this version, only a source from a trustworthy source known at the time of flashing the firmware running on the box can be flashed via AVM's standard web interface.

In order to understand what the last sentence means in concrete terms, let's briefly look at the basic principles of digital signing (for in-depth information, please refer to numerous articles on the Internet, e.g. **this** Wikipedia article).

As part of digital signing, the following steps/issues are essentially important:

- The creation of a key pair consisting of a private and a public key. The private key is given a password so that the key can only be used by the person who has the key and knows the password.
- The private key of the key pair is used by the sender of a digital message and serves the purpose of providing this digital message with a digital signature.
- The digital signature allows the recipient of the message to use the public key (also called the verification key) to verify the non-deniable authorship and integrity of the message.

For our application case, it means the following:

- The digital message is the firmware image.
- The sender of the message is the source from which the firmware image originated. The source must have the entire key pair (i.e., both keys) and know the password that protects the private key.
- The recipient of the message is the firmware version running on the box at the time of the flashing process. It must be in possession of the public key (the verification key).

A (digitally signed) firmware image is classified as "coming from a trustworthy source" if the public key of this source is known to the firmware running on the box and the signature verification process is passed.

For understandable reasons, we do not have the private, secret key with which AVM signs the original images (and even if this were different, we would also have to know the password for it). This leaves us with no choice but to try to create a self-signed image and force the firmware running on the box to accept it. The following obstacles must be overcome:

- The most difficult part from a mathematical/technical point of view is to understand what exactly does a signing process or a signature verification process consist of in the AVM firmware? Fortunately, the developer **PeterPawn** (very well known in the IPPF forum) has done a great job in this regard and **documented** everything as part of his **YourFritz project** and provided the corresponding **source code**, which is now also built into Freetz.
- As mentioned several times above, for a signed image to pass the verification process, the firmware running on the box must know the source's public key. Of course, this is not the case in regard to our self-signed image. This means we have to somehow manage to get our public key into the box. Once we've done this, each additional self-signed image can be flashed via the regular AVM web interface, provided you use exactly the same key pair for signing and don't forget to include the public key in each new image.

Strictly speaking, the second task is not entirely trivial. A separate article on this topic is to be released in the future. At this point, we can briefly suggest a few possible solutions:

- A downgrade (via recovery) to an older firmware version that still accepts unsigned images. From this older firmware version, a newer firmware version must then be flashed that contains our public key.
- If you happen to have Telnet access to the box, you can use the "mount over it" method (**mount -o bind ... ...**) to temporarily replace one of the AVM keys with your own.
- For NOR boxes, an image containing the public key can be brought onto the box using **push_firmware**.
- For NAND boxes, an image containing the public key can be brought onto the box using the **eva-to-memory method**.

Specific application in Freetz

- Activate the expert view in Freetz (“Level of User Competence” = Expert).
- Under "Firmware packaging (fwmod) options" activate the "Sign image" option and enter the password for the private key directly below (the password is removed from the version of the .config copied into the image).
- Next, build a Freetz firmware as usual.