Chapter 6

(Un)trusting firmware and the host OS

Firmware considerations

We would like to treat most of the platform firmware as untrusted. This applies to the Intel ME, other devices, and the BIOS. While it should be obvious why Intel ME should be considered untrusted, it's also prudent to treat the BIOS as untrusted even if we decided to use an open-source implementation, such as coreboot [14]. This is because the task of creating a truly secure, i.e. attack-resistant, BIOS implementation for Intel x86 platform seems like a very challenging task. Not to mention that it is currently very difficult (impossible?) to have a truly open source BIOS which would not need to execute Intel-provided blobs such as the Intel FSP.

The trick of keeping the platform's firmware on the trusted stick is a game-changer here, because we can be reasonably confident the stick will: 1) implement proper read-only protection, this way stopping any potential flash-persisting attacks originating from the platform, and 2) even if the firmware was to be somehow malicious, the construction of our stateless laptop leaves no places for the malware to store any data stolen from the user. (It could still try to leak it through networking, a problem we discussed in more detail in the previous chapter.)

There are two important exceptions with regard to trusting the firmware though:

1. If we decided to use an internal disk, as discussed earlier, then we would need to trust the disk's firmware to properly implement encryption, and read-only protection for select sectors/partitions,