2. If we decided to use the Embedded Controller (again, let's not confuse this with the Intel ME) to implement internal networking proxy (as discussed below), then we would need to trust its firmware also.

Of course, as already discussed, both of these devices would be fetching the firmware from the Trusted Stick.

Host OS considerations

It's tempting to also assume the host OS could be treated as untrusted, using the similar argumentation we just used to convince ourselves we didn't need to trust Intel ME or the BIOS...

Indeed, at least for the networking scenarios #0 (air-gap) and #1 (closed network of trusted peers), as discussed in the previous chapter, that might indeed be a justified assumption.

However, for the more open networking scenarios #2 and #3, this might no longer be the case. Indeed, an insecure OS might allow malware infections that could now use all the convenience of a locally-executing program to steal user data, collect additional personal identifying information, and exfiltrate all this to some remote server using one of the million of ways how modern malware typically would do that. This would naturally lower the bar for the adversary significantly, almost negating the benefits of a stateless laptop...

This means it is still prudent to run a secure OS on the stateless laptop.

Reconsidering BIOS and ME (un)trusting?

An alert user might, however, now point out that we cannot assume the host OS to provide any security if we don't trust the BIOS, or ME. In theory this is true, of course. In practice, however, we should consider how a malicious ME or BIOS could potentially inject malware into our (otherwise secure) host OS.

The only way for such an infection to occur would be either for the Intel ME, or the BIOS, to inject malware into the host memory. In practice this means that Intel would release a processor which, under certain circumstances (yet not depending on any persistent state) writes malware to the host memory pages.