On the applicability of software attestation techniques to embedded systems.

Politecnico di Torino
2019
December
Master's degree in Computer Engineering

Candidate: Marco Zudettich
Supervisors: Prof. Cataldo Basile, Prof. Antonio Lioy, Alessio Viticchiè

Software has been developed around any aspect of technology in the last few years. It has become a core part of every device, not only for personal use but also in industrial and automotive environments. A direct raise in security issues has come with its spread, particularly for programs executing in third-party contexts. A company's income might rely on software operating on devices it does not own (take smartphones for example). If physical control is not an option, controlling the device integrity might not be feasible. An attacker can open the device, attach a debugger and potentially tamper its software. In this case, the only possible defense is to detect these attacks and take the proper countermeasures.

Software attestation is a technique which tries to solve this problem by monitoring the integrity of a program remotely. There are multiple methods to check software integrity. One option is to have hardware dedicated to this function. This solution, proposed by the Trusted Computing Group (TPM)[1], relies on special secure hardware dedicated to cryptographic operations and key storage. Since additional components imply a higher cost and integration with the software, this method was excluded from this analysis. The other option is to use fully-software solutions. These solutions are less protected against tampering but they do not require additional hardware. A valid example of software-only attestation procedure is the ASPIRE[2] project's remote attestator. ASPIRE offers a complete framework for binary protection. It includes obfuscation and anti-tampering techniques. It is possible to create a completely functional and protected binary starting from the source code. The project contains a module that performs an integrity check over some areas of memory chosen by the user. There are two components required to execute this attestation technique: the client and the server. The client's portion of the attestator is responsible for performing a checksum over the protected memory areas. The only areas which can be protected are the ones remaining constant. Typically only part of the

[1] `https://trustedcomputinggroup.org/`
[2] `https://github.com/aspire-fp7/framework`

code present in the text segment is checked. The server side is responsible for controlling the validity of the checksum computed by the client. In this schema, only the server component can reside in a trusted environment, while clients are assumed to be vulnerable to tampering.

This attestator has some problems in terms of portability. One reason is the extensiveness of the ASPIRE framework. The project offers lots of different protection techniques, but every protection comes with a cost in performance. This is a problem especially for IoT devices, where computational resources are limited. The other reason can be found in the dependencies the attestator has, in particular, the use of the Diablo toolchain[3] and the OpenSSL[4] library.

This paper elaborates on two questions, both related to the feasibility of using fully-software attestation techniques in less powerful platforms, such as embedded systems. The first regards the possibility of solving the ASPIRE remote attestator's portability problems. A standalone test version of the remote attestator was extracted from the project. This detachment isolates the attestator from the whole framework and it also removes the Diablo toolchain's dependency. Thus, the portability increases. This extraction is not a trivial task. The Diablo toolchain is responsible for some binary modifications applied after the linking process. Emulating the same procedure requires a good understanding of the ELF file format's inner structures. A custom script was created to replace this functionality. The second point is an analysis of the extent to which this attestation procedure can be ported to different platforms. The attestator's requirements were discussed and analyzed during this work. In particular, some libraries to substitute OpenSSL were examined. WolfSSL[5] and Mbed TLS[6] were analyzed as two possible replacements. The results suggest that the support for these two libraries is extensive. About 25 embedded operating systems were investigated. Out of the top ten, eight of them potentially support at least one of these libraries. By adapting the attestator to them, porting it to most of the embedded operating systems on the market is attainable.

There is one last factor that needs to be considered. The attestator alone does not grant effectual protection of the binary. If there is no obfuscation, reversing its functionality and bypassing its checks might become a trivial process. The Diablo toolchain was in charge of obfuscating the binary. By removing it, the program is left unprotected. Some obfuscation mechanisms and tools were analyzed during this work to overcome this issue. Tigress[7] is reasonably one of the best instruments available in this regard.

In conclusion, it is important to remind the intrinsic vulnerability in systems not protected from physical tampering. Tampering can be made extremely difficult by using complicated techniques, such as the one analyzed during this thesis, but, given enough time and resources, any protection mechanism is bypassable. In general, this study shows the attestation procedure is portable to most of the embedded systems. This factor contributes to make it a remarkably valuable technique. If used correctly, it can cooperate with additional layers of protection to deter an attacker long enough to make him give up.

[3] `https://diablo.elis.ugent.be/`
[4] `https://www.openssl.org`
[5] `https://www.wolfssl.com/`
[6] `https://tls.mbed.org/`
[7] `http://tigress.cs.arizona.edu/`