Research Project Proposal: Interruptible Remote Attestation

DAVIDE LI CALSI, DAVIDE.LI@MAIL.POLIMI.IT

1. INTRODUCTION TO THE PROBLEM

The recent increase in the number of low-end MCUs being employed in various applicative scenarios in undeniable. While they certainly provide several benefits, their inherent simplicity results in a clear loss in their security capabilities. This is mostly due to their scarce results, that is limited computational power, lack of key hardware components and abstractions such as a MMU, few KBytes of memory... Implementing sophisticated security mechanisms is rather unfeasible, thus opening to a wide range of challenges. In that context, we dwelled on Remote Attestation.
Remote Attestation[7] is a process through which a remote Verifier can attest that a device (Prover) is in a legitimate state (i.e., uncompromised). The process is based on a challenge and response technique to ensure the freshness of the response and thus avoid replay attacks. The challenge usually consists in the computation of a HMAC over the entire Prover's memory. One can also choose what to attest. Some approaches, known as Static RA, only attest the Program Memory, while Dynamic RA[4, 10] attests also Data Memory and runtime integrity. The latter is much harder, since it requires the attestation of dynamic properties that change frequently and can take values in a huge space.
RA is particularly suitable for low-end MCUs. Since defending them is much harder than usual, RA could allow to spot an infected device and take the most appropriate action. That is relevant because these devices are used in a lot of critical applications, from smart devices to healthcare monitoring systems. Imagine a scenario in which an attacker has infected a medical device that measures some vital signs. This attack could possibly alter the measurements, resulting in a significant bias in medical diagnosis, possibly leading to severe consequences for the health of the patient. This example, and many other, show the relevance of RA as a whole.
Some approaches to RA require disabling interrupts before its execution. In fact, if an attacker has taken over the device, it might use spurious interrupts to interfere with the process and avoid detection, e.g., by self-deleting before the attestation routine can attest its memory region or by self-relocating to a new memory area that was already attested. That is why most state of art RA schemes require interrupts to be disabled for the entire duration of Prover-side calculations.
Unfortunately, for devices operating in real-time contexts, interrupts are a key feature that should not be disabled for too long. Thanks to some empirical evaluations[5], that were also confirmed by our own experiments, attesting small memories (few MB of memory) takes hundreds of milliseconds. Time-critical applications cannot tolerate being suspended for that long. For that purpose, some RA schemes that allow (possibly partially) interrupts were devised (more in Section 2). In spite of that, they suffer from some unpleasant pitfalls, or are simply not applicable to low-end devices, due to their extreme scarcity of resources. We believe the time has come to find some method that allows RA to be fully interruptible, allowing the Verifier to query the Prover's state without interfering with its activity, and that can work even on the most constrained device.

2. MAIN RELATED WORKS

Some RA schemes, such as memory locking and shuffled measurement attestation, allow interrupts but come at a cost.
In particular, memory locking requires locking already attested memory areas which might be a significant drawback depending on the applicative scenario. In this context, locking means making it read-only. Tsudik et al.[2] investigated the effectiveness of memory locking, defining multiple locking patterns that vary the locked blocks and time of locking. However, memory locking requires an underlying microkernel with a MMU, and is therefore inapplicable to low-end MCUs.
Shuffled-measurements-based (SM)[3][1] RA provides probabilistic guarantees on malware detection which might greatly vary. It prescribes that memory is attested in random order, according to a random permutation of the blocks in which memory is divided. The probability of detection depends on the information that the malware has about the memory to be attested. The best-case scenario allows for a detection probability of $1 - e^{-1} \approx 63\%$.
Unrelated to interuptible RA, we have examined a great number of researches that look into malware detection using HPCs[6, 9, 8]. These counters are usually included in the PMU of some CPUs, and allow to count some low-level architectural events. These works show how to perform malware detection by using HPCs on different platforms. Some works use a Database of benign HPC values to perform classification, more recent ones rely on Machine Learning.
Although RA and HPC-based malware detection have been two separate research paths, we believe in the possibility to reconcile the two.

3. RESEARCH PLAN

Considering the context of low power microcontrollers with a trusted execution environment, we believe there is space for new tools and techniques to allow for interruptible RA. In particular, we set out to investigate whether a trusted computing base can be used to attest not only memory but also the event history (both application-level and architectural-level) that happened while doing attestation.
In fact, we noticed that some Cortex-M low-power MCUs are natively equipped with hardware to count events and trace the high-level execution. Like many works have shown before us, low-level Performance Counters can be used for malware detection. These techniques are able to detect malware activity in general, i.e. they are designed to comply with a wide range of attacks. Therefore one can try using the same technique to detect transient/self-relocating malware during the execution of an interrupt handler. If that is possible, the counter vector obtained during the interrupt service can be used to determine if some malware has tampered with memory during attestation. By working on RA on low-end MCUs, advantages and challenges are brought up. This context implies you are dealing with a fixed, limited range of attacks, with well known characteristics. We believe that this could allow to achieve high accuracy even with simple models, thus saving a lot of computational resources at Verifier side. The classification performance could also be boosted by the fact that low-end MCU run few simple applications, thus limiting the variance in the possible legitimate behaviors. Furthermore, to the best of our knowledge no state of

art paper analyses the high-level activity of the device. Yet, we believe that the high-level events, if properly secured, can provide rich information to discriminate malicious activity. Because we work with simple MCUs that run a limited number of applications, we expect to successfully devise some high-level event counters. This will surely prove easier compared to more sophisticated general purpose CPUs, that need to deal with numerous and more complex applications.
Overall, two phases are required:

- **Offline phase**: a profiling phase that should happen before the deployment of the device, perhaps conducted by the vendor. This task consists in running benign applications on the target device, always monitoring the selected Performance Counters. Meanwhile, few malicious malware relocations should be emulated. Both these activities are meant to collect data to train a classifier, which will later run on the Verifier.
- **Online phase:** when the Verifier sends an attestation request, Prover replies with the attestation reply and the signed value of the collected counters. The Verifier should then feed the counters vector to its classifier, in order to determine whether the attestation response can be trusted or not.

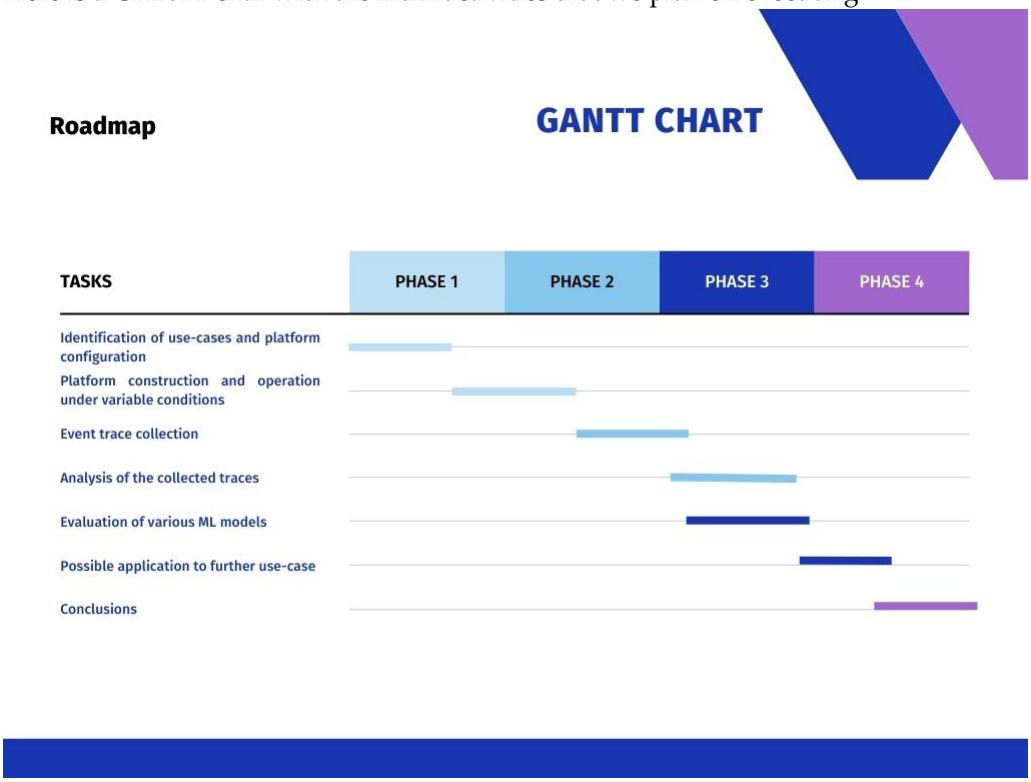
Due to the nature of the Performance Counters, we expect them to introduce a negligible Prover-side computational overhead w.r.t. to an existing non-interruptible RA scheme. In fact, enabling, disabling, and collecting them can be done by reading and writing few registers, an operation that corresponds to few instructions. As for the Verifier, the computational load greatly depends on the Machine Learning model that it runs. The scientific literature generally assumes that the Verifier has higher computational power than the Prover, so it is not strictly necessary to reduce its computational burden. Nevertheless, we will test and evaluate several models, with increasing capabilities, in order to find the least powerful model that can still achieve an acceptable accuracy given some scenario and external requirements.

We have planned the following roadmap to conduct future research activities:

- **Identification of two or more use-cases and associated platform configurations:** this task was already partially completed. We have constructed a small node, consisting of a Cortex-M33 on a ST-Nucleo-L552ZE-Q board. This node is equipped with a BLE module and some Temperature and Humidity sensors, in order to emulate the behavior of a smart sensor employed in an industrial platform. We also believe that healthcare could be a valid use-case, as several medical devices employ low-end Cortex-M MCUs.
- **Selection and construction of a use-case environment allowing variability of conditions:** we will try to evaluate the possibilities and limits of our methodology under variable operative conditions. We plan to examine its performance with varying network conditions, varying number of tasks, varying good state/compromised code
- **Analysis and classification of event traces**: we will try to contruct a more general methodology that describes which counters are more meaningful to detect a self-relocating/transient malware. Furthermore, we plan on including high-level counters in the analysis, i.e. counters that measure the rate of occurence of high-level events. Using the data generated by the platforms (constructed in the previous steps), we plan on training some Machine Learning models, from Logistic Regression to Deep Neural Networks.

- **Potential application to other use cases**: once all the previous points have been set and completed, we will try to extend the capabilities of our methodology. The ideal goal would be to obtain a flexible model that can be used to classify event traces coming from a broad category of devices. We will investigate this issue, and try to come up with the most general model possible.

Here is a GANTT char with the main activities that we plan on executing.



This research will require potentially new tools and methods to attest event history which might require, among other things, the characterization of valid event signatures and classification of real-life ones. The goodness of the devised attestation method can be measured in probabilistic terms and, as this is an initial proof of concept, we consider a detection probability quantitatively similar (+/- 5%) to best case SMRA to be a success.

REFERENCES

[1] AMAN, M. N., BASHEER, M. H., DASH, S., WONG, J. W., XU, J., LIM, H. W., AND SIKDAR, B. Hatt: Hybrid remote attestation for the internet of things with high availability. *IEEE Internet of Things Journal 7*, 8 (2020), 7220–7233. 2

[2] CARPENT, X., ELDEFRAWY, K., RATTANAVIPANON, N., AND TSUDIK, G. Temporal consistency of integrity-ensuring computations and applications to embedded systems security. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (New York, NY, USA, 2018), ASIACCS '18, Association for Computing Machinery, p. 313–327. 2

[3] CARPENT, X., RATTANAVIPANON, N., AND TSUDIK, G. Remote attestation of iot devices via smarm: Shuffled measurements against roving malware. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* (2018), pp. 9–16. 2

[4] DESSOUKY, G., ABERA, T., IBRAHIM, A., AND SADEGHI, A.-R. Litehax: Lightweight hardware-assisted attestation of program execution. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2018), pp. 1–8. 1

[5] ELDEFRAWY, K., RATTANAVIPANON, N., AND TSUDIK, G. HYDRA: hybrid design for remote attestation (using a formally verified microkernel). *CoRR abs/1703.02688* (2017). 1

[6] KADIYALA, S. P., JADHAV, P., LAM, S.-K., AND SRIKANTHAN, T. Hardware performance counter-based fine-grained malware detection. *ACM Trans. Embed. Comput. Syst. 19*, 5 (sep 2020). 2

[7] KUANG, B., FU, A., SUSILO, W., YU, S., AND GAO, Y. A survey of remote attestation in internet of things: Attacks, countermeasures, and prospects. *Computers Security 112* (2022), 102498. 1

[8] KURUVILA, A. P., MENG, X., KUNDU, S., PANDEY, G., AND BASU, K. Explainable machine learning for intrusion detection via hardware performance counters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022), 1–1. 2

[9] ROSSO, M., RENES, J., VESHCHIKOV, N., ALVARENGA, E., AND DEN HARTOG, J. Actionable malware classification in embedded environments using hardware performance counters. SPACE 2021: Eleventh International Conference on
Security, Privacy and Applied Cryptographic Engineering, SPACE 2021 ; Conference date: 10-12-2021 Through 13-12-2021. 2

[10] SUN, Z., FENG, B., LU, L., AND JHA, S. Oat: Attesting operation integrity of embedded devices. In *2020 IEEE Symposium on Security and Privacy (SP)* (2020), pp. 1433–1449. 1