The paper *"**Mind the Gap: Studying the Insecurity of Provably Secure Embedded Trusted Execution Architectures**"* [Oakland'22] performed a security analysis of two roots of trust for low-end MCUs, one of which is VRASED. While the paper refers to its findings as "insecurity of provably secure architectures", reported issues lie outside VRASED verified TCB and stem from:

(i) axiomatized requirements of VRASED that were not observed in one particular exemplary prototype that integrates VRASED with the openMSP430 core.
(ii) unverified software components from forks of the VRASED project that modify its implementation.
(iii) inobservance of VRASED threat model which rules out hardware attacks. Specifically, one attack that modifies ROM-resident code, and one attack that adds a trusted yet malicious peripheral to the MCU internal bus.

We appreciate the fact-checking of axioms - item **(i)** above - for the openMSP430-based prototype. We are also strongly in favor of offensive security as a method to make systems more robust which is the main lesson in the "Mind the Gap" paper. We have also integrated applicable suggestions into the openMSP430-based instantiation of VRASED. Nonetheless, we also believe that the use of the verbiage "insecurity of provably secure architectures" and "vulnerabilities in VRASED" may be misleading. It is important to note that VRASED is a Remote Attestation (RA) module/architecture. Regardless of whether the RA architecture is correct and verified in itself, assuring its correct instantiation to each particular core is a separate effort. VRASED includes a list of axioms: requirements that must be observed in the integration. The fact-checking of these axioms is required whenever one decides to deploy VRASED on a different platform. The integration of VRASED with particular MCU cores has not been our focus and was never claimed provable as it would make the verification results core-dependent.

The preceding discussion does ***NOT*** mean that there are no vulnerabilities within the logic of VRASED verified implementation. They may very well exist! Verification should not be interpreted as unconditional security. However, the specific set of reported issues do not consist of vulnerabilities within VRASED RA module and none of them falsify provable features, but only axioms. Simply put, fixing them requires changing the openMSP430 integration but not VRASED verified RTL or software components. The latter is what is verified, not the former.

While we believe this is already clearly discussed in the VRASED paper, below we revisit the extent of VRASED verification (Section 1). Section 2 provides a discussion of the issues reported in the "Mind the Gap" paper. Our view was shared and discussed with the authors of the "Mind the Gap" paper. However, since we did not have the opportunity for a public response (the first contact was after the "camera-ready" version of the "Mind the Gap" paper was already released) we are making the conclusions public.

Section 1. The Extent of VRASED Verified RA Architecture

Claims about "insecurities in VRASED" or "insecurities of provably secure architecture" indicate an important misunderstanding about what exactly constitutes VRASED verified RA module:

- **VRASED is not the entirety of its open-source Github repository.** The repository contains an example prototype of the integration of VRASED with an open-source MCU core: the openMSP430.
- **VRASED verified RA architecture is ONLY what is inside the VRASED sub-directory in this repository.** The integration of the VRASED RA architectecure (the verified RA implementation inside the VRASED directory) with the openMSP430 core is done only as an example. Its

purpose is to allow testing and to enable measuring relative hardware/software overheads with respect to a low-end MCU core. VRASED verification is not geared to a specific core.

- **VRASED implementation is verified for a generic machine model, i.e., axioms,** that inform developers (interested in integrating VRASED with a specific core) what requirements, e.g., inputs and memory configurations, must be observed for a secure integration with VRASED. Having a general machine model is important because the internals of different cores work differently. To make this point clear, not even different MSP430 models would have the same memory layout. A real MSP430 core is not necessarily consistent with the openMSP430 implementation. Hence, making any verification effort openMSP430-specific would limit its portability and practicality.

VRASED is a verified RA module based on a generic machine model. VRASED paper notes that the purpose is not to create a core-specific RA architecture (see the Scope section). Indeed, one of the most appealing properties of hybrid RA architectures (similar to practical hybrid architectures such as secure boot) is core-independence, i.e., no need to change CPU internals, such as the ISA. Ideally, porting VRASED across different MCUs should be relatively effortless and secure, **as long as all VRASED axioms are observed (i.e., fact-checked) for that particular integration**. However, exhaustively fact-checking VRASED axioms with respect to the toy openMSP430 prototype has not been our focus.

We note that in no part of the VRASED verification effort, have we claimed to prove the machine model axioms. Such a claim would be in itself contradictory, since axioms, by definition, represent non-provable statements in a formal system. When we state something as “an axiom”, we are explicitly pointing out that such a requirement has not been proven. One of the points of formally specifying axioms is to have a list of non-provable requirements that one needs to observe when instantiating the architecture. VRASED paper explicitly rules out the verification of the underlying core.

Section 2. Responses and Clarifications to the Reported Items

[Item codes are based on their original indexes in the “Mind the Gap” paper]

[B1] Correct passing of DMAaddr is axiomatized, hence not provable. Passing of this signal is core-specific. For the openMSP430-based prototype, we have never integrated or tested DMA signals. The openMSP430-based prototype has been amended to implement the correct input passing. However, this issue lies outside VRASED verified architecture.

[B2] Once more, correct assignment of memory regions is axiomatized, therefore it is not provable. It is a part of the integration with each particular MCU core. Hence it is not in the scope of VRASED architecture or verification thereof.

[C1] Memory reads/writes are axiomatized in **Definition 5** in the VRASED paper. Once more, any core that does not adhere to this requirement does not benefit from VRASED/APEX provable guarantees. In addition, **C1** makes hardware changes to the MCU internal bus. Such a possibility is ruled out by VRASED’s threat model. In particular, **C1** modifies the peripheral bus behavior in hardware to violate the (required) machine model.

[C2] The proposed [C2] attack requires re-compiling SW-Att with specific flags to remove this memory zero-ing. Hence the attack violates the threat model, because (by assumption) ROM can not be overwritten after manufacture-time. After a discussion involving the authors of both papers, we have identified a difference of opinion regarding the meaning of VRASED’s “compiler’s semantic preservation”

assumption: VRASED authors consider memory side-effects such as writes to memory addresses a part of semantic equivalence/preservation between C language and CPU instructions. The authors of the “Mind the Gap” paper do not.

As to the specific reported issue: the initialization of the stack pointer outside SW-Att is intentional to keep low-level unverified code outside the TCB. In the same way, the zero-ing with the size of the MR region (upon an SW-Att call that could leak data through MR) is also intentional. In that way, any other values for the stack pointer will cause a reset before any leakage.

Ideally, this should have been captured in Lemma 2 in VRASED paper, but it is unclear how to specify it in LTL. For that reason, we intentionally introducced zero-ing of memory in VRASED code. Therefore, removing it from the ROM implementation will clearly result in an issue.

[C3] We completely agree that any secret-dependent implementation of **memcmp** would be insecure. In fact, the secret-independence requirement for SW-Att implementation is stated in the VRASED paper.
While the “Mind the Gap” paper claims to exploit a vulnerability based on a secret-dependent implementation of the **memcmp** function, no implementation of **memcmp** exists or was ever released as a part of VRASED’s architecture or paper.

Looking at the associated GitHub pull request, the vulnerable code was taken from the RATA project and was conflated with VRASED to claim having found three (**[C3, C4, C5]**) “vulnerabilities **in VRASED**”.
There are a few important points to be made here:

- RATA is a project distinct from VRASED. RATA repo is managed by a different collaborator who was not involved in VRASED design/paper. It should not be conflated with VRASED.
- At the time of writing, RATA repo was still under construction and the code therein wasn’t even functional when it was taken.
- RATA paper is not about verification of authentication, neither is the VRASED paper. Appendix A of VRASED paper simply discusses how authentication could be supported and what would be the LTL properties required from the hardware to support that. It does not tie into VRASED end-to-end proofs. Not even the protocol logic is verified – it is taken as-is from reference [10] in VRASED paper. Authentication of the Verifier was not claimed as verified or released as a part of VRASED repo.

[C4] and [C5]: as far as we could tell, these work based on **[C3]**, which doesn’t apply to VRASED TCB.

[D] We agree that any system that runs software or proves properties about running software (even if the proof is computed by hardware) requires that assumption. From another perspective, attestation is an authenticated picture of memory, if the true behavior of instructions as implemented by the hardware is not known, the attestation result is meaningless. While axioms [A1] to [A7] are informal and non-exhaustive, we hope they convey the non-trivial assumptions (VRASED paper states "we assume the MCU architecture strictly adheres to, and correctly implements, its specifications" even before listing [A1] to [A7]). Arguing completeness of a given set of axioms to model a real system is hard. We hope that [A1] to [A7] contains the non-trivial relevant subset.
Nonetheless, the “Mind the Gap” paper discussion in item D is clearly in-line with our view. It discusses a hypothetical core that may not adhere to VRASED requirements and what could happen in this case. This argument is sound for both hypothetical and real cores. This is the exact reason to axiomatize requirements as much as possible. When an axiom is not observed, the inconsistency that leads to insecurity is **not** “in the provable architecture”. The inconsistency lies in non-provable parts external to the architecture. Identifying these axioms is one of the benefits of formal specifications.