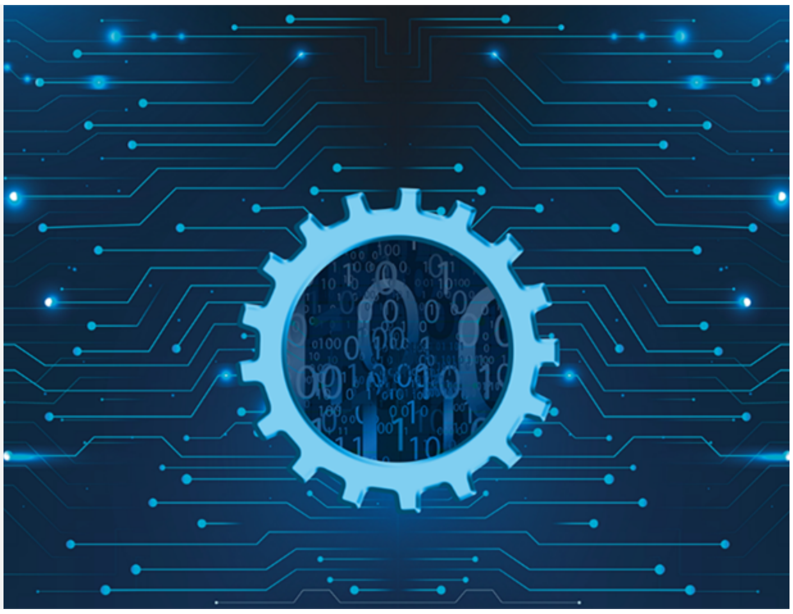
SUCCESS STORY 2022/1

DEPS - https://deps.scch.at
Dependable Production Environments with Software Security

Host: SCCH, www.scch.at

Programme: COMET – Competence Centers for Excellent Technologies

Programme line: COMET-Module



IP PROTECTION USING SELF-MODIFING CODE

PREVENTING PRODUCT PIRACY THROUGH SOTWARE-HARDWARE BINDING

Industrial-scale reverse engineering is a serious problem, with estimated annual losses for industry at 6.4 billion euros in Germany alone[1]. Typically, the main effort needed to steal the intellectual property (IP) of companies that make sophisticated products with software components, resides in replicating the hardware. Instead, software can often be copied verbatim with no reverse engineering required.

In DEPS we investigate new approaches to prevent the described IP theft. A key development in this sense is the idea of binding (or gluing) programs to hardware, so that they only execute correctly in the target machine. In this way, if a protected program runs on a machine other than the intended one, then it will behave differently.

To achieve this vision in a way that makes reverse engineering of the protection a daunting, almost impossible task, we have developed a sophisticated approach based on physically unclonable functions (PUFs) and self-modifying code.

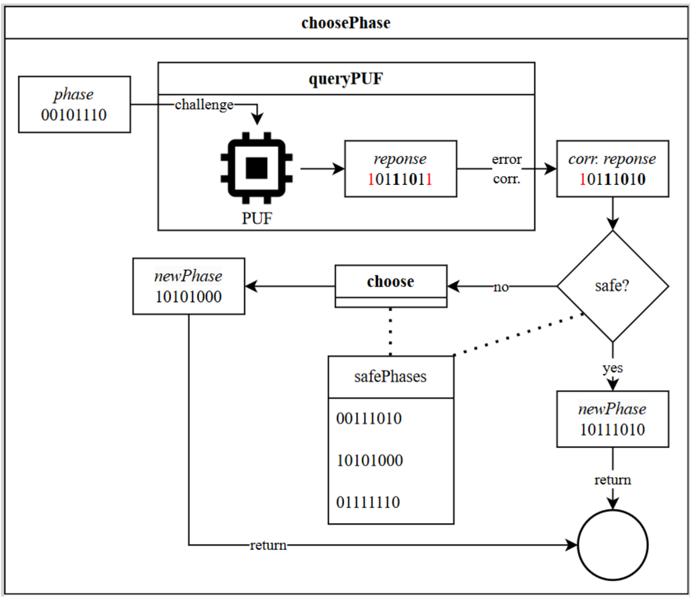
PUFs are hardware-based security primitives which cannot be cloned, since they depend on minor, unavoidable variations in the manufacturing process of hardware components. In turn, self-modifying code are reflective algorithms that can change their behavior at run-time. Putting both together, we can make a program behave correctly only in the target hardware. The key to prevent reverse engineering of the correct code is to only instrument the correct behavior at run-time and protected by PUFs.

[1]www.vdma.org/documents/34570/51629660/VDMA+Study+Product+Piracy+2022_final.pdf

Research Question

Can our envisioned copy-protection mechanism be such that:

- Efficient and secure protection is provided to a wide class of industrial software.
- Programs only execute correctly on their target hardware.
- Programs still run on other machines but behave differently and in ways that are less obvious to detect.
- Hardware-software binding is based on any commercially available PUF.
- Reverse engineering of the protection is unfeasible and uneconomical.
- Critical properties, in particular operation safety, are preserved in the protected software irrespective of where it is executed.



DEPS Scientific Answers

We have developed a theory of reflective algorithms [1] which enables us to formally specify the required behavior of self-modifying code, step-by-step and at any required level of abstraction via Abstract Sate Machines (ASMs). This is complemented by a logic which unlocks reasoning about reflective code and formal proof of properties such as safety.

Using reflective ASMs we have modelled and precisely specified in [2] a universal approach for hardware-software binding, by means of which security, in particular copy protection, can be supported in a proper and consistent way.

A refinement of the model in [2] resulted in a practical approach to copy-protection that works seamlessly with a wide class of control state algorithms and satisfies the requirements in our research question [3]. In this approach, the different phase changes (or equivalently state changes) are governed by PUF responses, and only correct in the target machine (see the schema). Unintended behavior preserves critical properties (e.g., safety) thanks to a clever use of symbolic execution.

Company partner Symflower was key in achieving this vision. We were able to efficiently define safe states using their symbolic execution technology.

In parallel, we have achieved a milestone with regards to grammar inference and model-based fuzzing [4] (recognized by a best paper award at MEDI 2022). We now have the tools needed to automate the process of determining program phases (states) as required by our protection approach [3]. This will be achieved by fuzzing-based analysis of program execution paths.

Related DEPS Publications

[1] K.-D. Schewe, F. Ferrarotti: Behavioural theory of reflective algorithms I. Sci. Comput. Program. 223: 102864 (2022)

[2] L. Tong, K. Xu, J. Hu, F.Ferrarotti, K.-D. Schewe: Exploration of Reflective ASMs for Security. ABZ 2023: 185-192.

[3] D. Dorfmeister, F. Ferrarotti, B. Fischer, E. Haslinger, R. Ramler, M. Zimmermann: An Approach for Safe and Secure Software Protection Supported by Symbolic Execution. IWCFS 2023.

[4] H. Sochor, F. Ferrarotti, D. Kaufmann: Fuzzing-Based Grammar Inference. MEDI 2022: 72-86

Contact:

Thomas Ziebermayr, DEPS Coordinator, SCCH, T +43 50 343 890, thomas.ziebermayr@scch.at

Consortium:

Federal Ministry Republic of Austria Climate Action, Environment, Energy, Mobility, Innovation and Technology

Federal Ministry Republic of Austria Labour and Economy

Austrian Research Promotion Agency
Sensengasse 1, A-1090 Vienna
P +43 (0) 5 77 55 - 0
office@ffg.at
www.ffg.at