Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 180 (2021) 862–866

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Industry 4.0 and Smart Manufacturing

Protecting Intellectual Property Rights of Industrial Software

Thomas Ziebermayr

Software Competence Center Hagenberg GmbH (SCCH), Softwarepark 21, 4232 Hagenberg

Abstract

The importance of software is increasing and software has become the main driver of innovation in many industrial products. The value of software is growing dramatically, which requires effective mechanisms to protect intellectual property rights (IPR). Current approaches are expensive by means of involved effort or run-time performance, or they not as secure as required. Recent research results show promising results in the area of precisely identifying hardware environments, software obfuscation, and prevention of disassembling and tampering. The vision of a secure, effective and easy to use software protection is the core driver for the research project DEPS. The vision is that industry can securely deliver software together with their hardware products without the risk of infringement of intellectual properties rights due to illegal copies, reverse engineering, and modification.

Keywords: IPR Protection of Software, Industrial Software, Automation

1. Introduction

Digitalization is changing businesses. Over the last decades, software has become the main innovation driver for high-technology products. Entirely new product developments have been achieved in terms of innovative software features and major advancements in hardware are relying on software as enabling technology. Today, thus, any product from small sensors to large industrial machinery contains a significant share of software and this share continues to grow at a tremendously fast pace [1].

Since software captures vital business know-how and intellectual property (IP), new threats have emerged for digitally intensive industries. Among them, intellectual property theft has turned into an increasingly critical issue, encompassing the theft of trade secrets, plagiarism of copyrighted materials, as well as violations of patents and trademarks. In today's globalized world, innovation in new materials or production methods is becoming more and more difficult. Industry is under enormous pressure by international competition. Companies and funding bodies invest huge amounts of money in research and development to remain competitive. Therefore, theft of intellectual property

10.1016/j.procs.2021.01.336

poses a major threat for the innovation capability as well as for the economic growth of many companies and, likewise, for the industry as a whole.

The impact of cyber theft of trade secrets from European businesses and organizations has been estimated to cause an overall economic loss of 83 billion Euro per year during the period 2013-2017 and a consequential loss of 671,000 jobs [2]. A particularly critical issue is product piracy, which has also been subject of a recent study [3] by VDMA, the German mechanical engineering industry association. In the last two years, 74% of companies in the domain of mechanical engineering and plant construction have been affected by product and brand piracy. More than half of these cases are related to plagiarism of products by illegal reproduction of components and even entire machines.

Software capturing pivotal intellectual property is particularly vulnerable to plagiarism. It is a key characteristic of software to allow exact copies at virtually no cost. Thus, essential parts of a product can be pirated simply by making illegal copies of the software. Furthermore, key knowledge and trade secrets encoded in the software can be stolen by reverse engineering algorithms and data.

In this paper, we give an overview of the state of the art in software protection (Section 2) and outline a protection approach based on software diversity to augment existing protection mechanisms (Section 3). Implementing and adapting this approach is subject to an ongoing research project. The status and an outlook on the work planned in this project is presented in Section 4.

2. Methods and Techniques for Software Protection

Attacks on software systems either come from outside (e.g., code/command injection, session hijacking, denial of service attacks) or from inside, also called man-at-the-end (MATE) attacks [4]. In a MATE attack, the attacker has full access and control over the compiled (binary) version of a software system. Therefore, the attacker can execute the software system, make copies, and use static or dynamic analysis, disassembling as well as debugging to reverse engineer and/or to modify the software (e.g., to remove copy protection mechanisms).

Methods to prevent MATE attacks are commonly known as software protection methods [5]. The main methods and techniques for software protection can be grouped into following categories [5]:

- *Tamper proofing* to prevent modifications to the software
- *Obfuscation* to prevent reverse engineering
- *Watermarking* to track programs and clones
- *Birthmarking* to identify copies of programs

Tamper proofing deals with preventing and detecting modifications to software. A typical approach is to check the integrity of the software, e.g., by using checksums or by monitoring execution paths. When modifications are detected, self-defense mechanisms of the software can be triggered. Self-defense can include, for example, refusing to execute, sending messages to the owner of the software, or self-repairing by restoring the original state. Tamper proofing is often combined with obfuscation, also to hide tamper-proofing methods.

Obfuscation methods and techniques change the data used by a program or the program itself at compile time (static rewriting) or at run-time (dynamic rewriting) to make it harder to understand and reverse engineer [6]. Common obfuscation methods and techniques include, e.g., reordering data and software structures, flattening hierarchies to obscure the control flow, inserting superfluous or dead code, and encryption of software to prevent static analysis. Program obfuscation is often used in combination with other protection methods and techniques [7], e.g., to protect the other protection methods and techniques such as code used for tamper proofing.

Watermarking allows to identify software by a secret information that is embedded in the executable program (static watermarking) or computed at run-time (dynamic watermarking). Generally speaking, watermarking embeds a hidden message to track copies of the software or parts thereof and it allows us to prove ownership in case theft has occurred [8]. Watermarking is widely used to protect media such as pictures or films in order to discourage piracy but it can also be applied to software programs.

Birthmarking (also known as fingerprinting [9]) is closely related to watermarking but uses a unique secret information for every instance of the software, e.g., a message containing the customer ID. Thus, this approach allows tracing copies of the software back to its initial source, i.e., the violator of the copyright.

Software protection is considered one of the most challenging problems in software security since attackers have full access to software and hardware [5]. At a certain level, it is even impossible to protect software from MATE attacks. However, software protection usually does not target the (illusive) goal of absolute security but rather increases the effort for the attacker to an unfavorable cost-value ratio. This can be achieved by advanced protection methods and techniques that are hard to bypass or by a combination of complementary methods and techniques. In the next section, therefore, we propose an enhancement of the outlined software protection methods and techniques, focusing especially on the domain of industrial control software part of industrial machinery, plant automation and embedded systems.

3. Vision of Enhanced Software Protection for Industrial Software

There is a strong need for effective, secure and easy to use solutions for software protection in industrial environments. Although several methods and techniques for software protection are available, the vision is to provide an extension to current approaches with the goal to strengthen the protection and to overcome their shortcomings.

First, in order to be practically applicable in the domain of industrial systems, we define requirements and constraints that new protection methods and techniques have to fulfill:

- Protect software effectively from disassembling or reverse engineering to keep intellectual properties such as algorithms and data safe in the software.
- Securely bind the execution of software to a defined hardware environment without the need for additional hardware keys, dongles, or hardware-based identification mechanisms.
- Protection mechanisms for the software should not have any negative effect on the execution performance, the run-time memory consumption, and the size of the deployed binary version of the program (due to storage space limitations and bandwidth consumption in over-the-air updates).
- Protection mechanisms have to be easy to apply, configure and administrate even in scenarios of mass production, mass customization, remote/online deployment, as well as frequent updates of software and/or hardware.

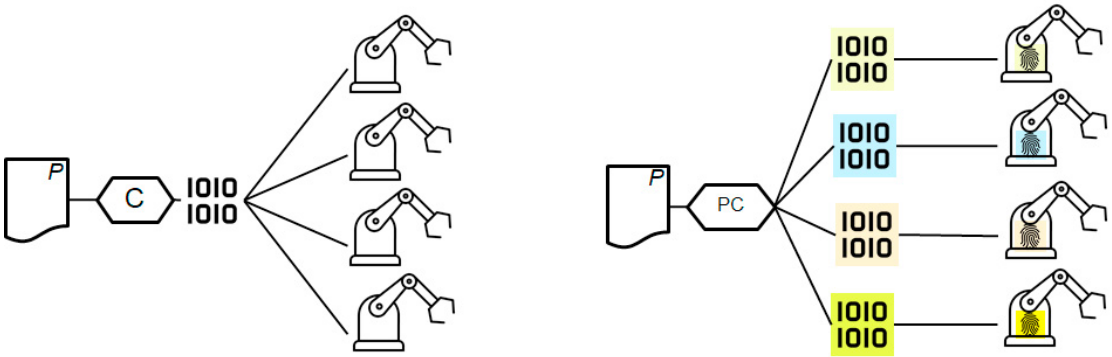

Fig 1: Comparison conventional industrial software (left) and enhanced software protection for industrial software (right)

Figure 1 (left-hand side) illustrates the current situation in provisioning software for industrial systems. Usually the software is compiled to an executable binary representation of the software. Copies of this binary are then deployed on the hardware devices, so every hardware instance has the same software binary installed. In this scenario, the software may be protected with different mechanisms as outlined above. However, once a successful attack has been found, it compromises every binary instance of the software and therefore all hardware devices. In addition, illegal copies of the software can be distributed as they run on each of the hardware instances.

The envisioned solution (right-hand side of Figure 1) addresses these weaknesses in two ways. First, the idea is to overcome the inherent software monoculture by generating diverse binary instances of the same software [10]. This can be achieved with a protection compiler (*PC*) that produces different binary representations yet with the same functional behavior from the original program *P*. These different software instances cannot be compromised by replicating the same attacks, so a single successful attack on one instance has less impact on the entire ecosystem. In addition, the generation of different binaries can also include and combine protection methods like tamper proofing and obfuscation for an enhanced combined protection that is customized to each individual instance.

The second part of the approach is about hardware identification. The software running on the hardware is able to identify the hardware instance where it runs by unique properties of the hardware system. The vision is to enable this identification without additional hardware support. The hardware properties can be tied into the program logic like array access or similar. This tight integration prevents the detection and makes it very difficult to circumvent. Therefore, the software can be made to only work correctly on the related hardware and illegal copies of the software are rendered useless.

The combination of both mechanisms together provide a comprehensive protection for industrial software:

1. It securely binds the software to unique hardware properties and therefore provides effective copy protection without additional hardware extensions.
2. The combination of diverse binaries with tamper proofing and obfuscation as well as the binding to hardware properties provides effective protection of the protection mechanisms themselves.
3. Optimized software generation and direct integration of protection methods in the software should avoid expensive outbound communication, key management or encryption of the software.
4. Automated analysis, generation and deployment of the software makes the protection approach applicable for mass production.
5. Updates to hardware and/or software require re-compilation of the software for the specific target hardware. However, these tasks can be automated with hardware/software management.

The research program Dependable Production Systems (DEPS) addresses the aforementioned vision. It focuses on the application of the protection methods in industrial environments where software manages production machines or logistic processes. The project starts with fundamental research on hardware identification mechanisms for software protection. In the research program, additional work will be conducted to enhance methods for secure software generation. The combination of protection methods with hardware binding will be the third step of the research agenda. Finally, the last step in the DEPS roadmap is to optimize protection methods for usage and application in industrial software production.

4. Conclusion and Future Work

There is an increasing need in industry for software protection as the investments in software are constantly growing and software nowadays represents a crucial asset for many companies. The complex software and software production processes including DevOps require protection methods, which are secure and easy to use. The ambitious vision is to provide a strong protection solution while addressing all requirements of industry with promising recent research results on software diversity and hardware identification. The roadmap to achieve this vision is outlined by the milestones of the ongoing research project DEPS.

Acknowledgements

The research reported in this paper has been partly funded by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), the Federal Ministry for Digital and Economic Affairs (BMDW), and the Province of Upper Austria in the frame of the COMET - Competence Centers for Excellent Technologies Programme managed by Austrian Research Promotion Agency FFG.

References

[1] Ebert, Christof, and Carlos Henrique C. Duarte. "Digital Transformation", IEEE Software. 35, no. 4, 2018, pp. 16-21.

[2] European Union Intellectual Property Office (EUIPO). "2020 Status Report On IPR Infringement: Why IP Rights are important, IPR infringement, and the fight against counterfeiting and piracy", European Union, June 2020, DOI:https://doi.org/10.2814/165063

[3] Verband Deutscher Maschinen- und Anlagenbau e. V. "VDMA Studie Produktpiraterie 2020", VDMA, 01.04.2020; available online: https://industrialsecurity.vdma.org/en/viewer/-/v2article/render/48531930

[4] Akhunzada, Adnan, Mehdi Sookhak, Nor Badrul Anuar, Abdullah Gani, Ejaz Ahmed, Muhammad Shiraz, Steven Furnell, Amir Hayat, and Muhammad Khurram Khan. "Man-At-The-End attacks: Analysis, taxonomy, human aspects, motivation and future directions", Journal of Network and Computer Applications 48, 2015, pp. 44-57.

[5] Falcarin, Paolo, Christian Collberg, Mikhail Atallah, and Mariusz Jakubowski. "Guest Editors' Introduction: Software Protection", IEEE Software, 28, 2011, pp. 24-27. DOI:https://doi.org/10.1109/MS.2011.34

[6] Schrittwieser, Sebastian, Stefan Katzenbeisser, Johannes Kinder, Georg Merzdovnik, and Edgar Weippl. "Protecting software through obfuscation: Can it keep pace with progress in code analysis?", ACM Computing Surveys (CSUR), 49, no. 1, 2016, pp. 1-37.

[7] Christian Collberg. "Code Obfuscation: Why is This Still a Thing?", Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy (CODASPY '18). ACM, 2018, pp. 173-174. DOI:https://doi.org/10.1145/3176258.3176342

[8] Dey, Ayan, Sukriti Bhattacharya, and Nabendu Chaki. "Software watermarking: Progress and challenges", INAE Letters, 4, no. 1, 2019, pp. 65-75.

[9] Collberg, Christian, and Clark Thomborson. "Watermarking, tamper-proofing, and obfuscation-tools for software protection", IEEE Transactions on software engineering, 28, no. 8, 2002, pp. 735-746.

[10] Larsen, Per, Andrei Homescu, Stefan Brunthaler, and Michael Franz. "SoK: Automated software diversity", Proceedings of the IEEE Symposium on Security and Privacy, IEEE, 2014, pp. 276-291.