Towards a Cloud-based System for Software Protection and Licensing

Andreas Schaad[1], Bjoern Grohmann[2], Oliver Winzenried[2], Ferdinand Brasser[3] and Ahmad-Reza Sadeghi[3]
[1]*University of Applied Sciences Offenburg, Badstr. 24, Offenburg, Germany*
[2]*Wibu-Systems AG, Rueppurer Strasse 52, 76137 Karlsruhe, Germany*
[3]*University of Darmstadt, Mornewegstrasse 32, 64293 Darmstadt, Germany*

Keywords: Software Encryption, Licensing, Industry 4.0, Cloud Infrastructure.

Abstract: In this paper we report on the commercial background as well as resulting high-level architecture and design of a cloud-based system for cryptographic software protection and licensing. This is based on the experiences and insights gained in the context of a real-world commercial R&D project at Wibu-Systems AG, a company that specialises in software encryption and licensing solutions.

1 INTRODUCTION

The protection of software, digital artefacts and intellectual property becomes continuously more challenging with the increasing interconnection of industrial components. Traditional approaches to protect software are the combination of encryption, code obfuscation and locally attached hardware trust anchors (often called “dongle”). This dongle provides required cryptographic material to en- and decrypt application code and data at runtime of a system. At the same time access to protected functionality of the software can be controlled to implement commercial licensing models.

However, this approach does not necessarily scale (technically and economically) in physically and logically distributed systems. The question is whether it is possible to provide dongle functionality as a set of cloud-based services.

On basis of the experiences gained in an early stage industrial proof of concept, we describe a set of requirements for such a real-time cloud-based software protection and licensing service. We discuss a possible corresponding architecture and resulting design decisions. The commercial implications of offering such a cryptographic cloud service are also addressed.

The presented work in this position paper will be further conducted and extended with concepts from the trusted computing domain in a national 3-year funded project “CloudProtect”. The goal is to build a highly scalable and secure cloud service that provides required cryptographic material in real-time to decrypt protected parts of a software stack.

We hope our short discussion can serve researchers as a scenario to further position their own research work.

2 BACKGROUND

The increasing automation in the industrial sector requires the protection (confidentiality and integrity) of software. This software could be the highly confidential algorithm for a laser cutting machine, configuration data of a welding machine or a digital blueprint required by a 3D printer. Specifically, where machines may run in countries with different approaches to protecting intellectual property, the owner or operator of a machine wants to protect and exercise control over such digital assets.

For that reason, approaches to protecting software and data against reverse engineering or tampering by means of encryption and obfuscation are widely used. Such technologies can at the same time enable the owner of some software to define license conditions for the user. Granting access to a specific part of a software is thus done with respect to security as well as commercial policies.

Schaad, A., Grohmann, B., Winzenried, O., Brasser, F. and Sadeghi, A-R.
Towards a Cloud-based System for Software Protection and Licensing.
DOI: 10.5220/0006899505320536
In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications (ICETE 2018) - Volume 2: SECRYPT*, pages 532-536
ISBN: 978-989-758-319-3

2.1 Traditional Technical Approach

2.1.1 Protecting Software

A vendor of software (ISV – Independent Software Vendor) encrypts his products before selling the software to end customers. In other words, the vendor of a machine would encrypt and/or obfuscate software that is sold in combination with a machine. Technically, the ISV will protect his software at build time with the help of specific commercial libraries available for most modern software stacks (C++, Java, .Net, …).

In a very simple scenario that means that a .Net or Java program is encrypted at a method level and the overall call stack is made aware of the fact that in order to execute such a protected method some cryptographic key is required. The vendor will also define the supported commercial licensing models (In the simplest case: Gold, Silver or Bronze versions of the same software). The end user should thus only be able to use the software according to what was commercially agreed and paid for.

This approach not only keeps data confidential as long as possible but also supports implementation of concepts such as “counters” to measure how often a functionality may be invoked.

2.1.2 Activating Software

Once such protected software is shipped to the end user or operator it first has to be activated. A set of unique cryptographic tokens is generated on basis of a fingerprint (by combining CPU, Operating System, Disk Size, …) of the customer system and transferred to the local trust anchor (i.e. either a physical dongle attached to a computer or controller of a machine or a software dongle hidden in the machine logic).

2.1.3 Using Protected Software

When the software is eventually used in production it will check whether it can be started at all, whether certain branches or functions / methods can be executed or how (often) some functionality can be executed (as defined in the commercial license agreement). Non-authorised invocation of functions will fail.

All such checks are done against a (external hardware) dongle that basically acts as a small cryptographic processor and key store. Though in essence a cryptographic (symmetric) key is decrypting code in real-time we refer to this operation as performing a “license check”.

2.2 Requirements

Though in larger on-premise settings, a licensing server will allow a group of users to work with protected software, there are still certain issues with such current traditional approaches:

- Physical dongles are tamperproof but if lost, the keys are also lost.
- Pure software-based dongles (and thus required decryption keys) are possible – but are significantly easier to attack (especially when a machine is operated in non-trusted environment).
- If an ISV wants to offer a pure cloud solution, maybe even in combination with a machine, the protection and licensing service should also be offered as a service (SaaS).
- On-premise solutions using local licensing servers to support groups of users or machines cannot be directly used in a cloud setting.
- Current on-premise licensing is based on very coarse parameters and it would be desirable to measure precisely how a software is used (in accordance with existing privacy regulations).

The introduction of a cloud-service for software protection and licensing requires to consider technical as well as commercial requirements. At the current moment we are not aware of any such cloud-based service and operational infrastructure.

2.2.1 Technical Requirements

A set of selected high-level technical requirements can be summarised as follows:

- The license service shall provide the required cryptographic material to allow a service consumer to decrypt software at runtime.
- The service consumer will interact in real-time with this cloud service and, if the commercial license supports this, receive the required keys to perform decryption.
- The software owner can define at which intervals a check is required, e.g. for certain applications even every 10 seconds or less.
- If a network connection is not available for a configurable time, usage of the software must still be possible.
- The cloud-based license server must be able to handle parallel incoming license validation requests at a high rate.
- Exchanged data must be secured at the transport and message layer.

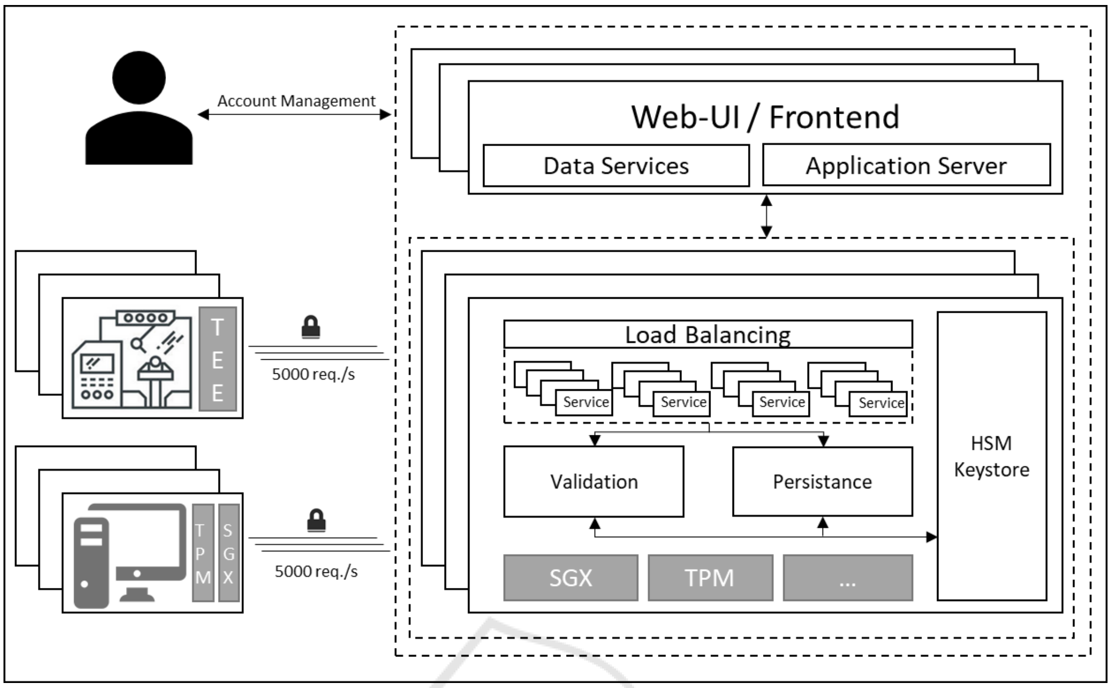

Figure 1: CloudProtect Architecture.

2.2.2 Commercial Requirements

The commercial requirements can be summarised as follows: The overall cost of offering such a service (pure technical costs as well as administrative) must be lower than the generated revenue. While this sounds trivial at first, we now have to deal with a situation where computation of cryptographic material is not done by a locally attached dongle (and thus consumption of local electricity) anymore.

We have to consider that in a cloud model "license checks" need to be performed for 1000s of end users or services – and the required CPU cycles now need to be paid for by the cloud operator.

This requires us to minimise technical and administrative costs if a cloud licensing solution should be offered at a price comparable to that of a traditional on-premise solution.

Offering this service without cannibalising the existing revenues of on-premise software protection solutions is a separate matter. However, we will need to find a suitable economic model to validate technical effectiveness of our architecture and implementation.

3 ARCHITECTURE

On basis of the discussed (selected) requirements we discuss a first sketch of an architecture.

3.1 Core Cloud Service

A CloudProtect instance exhibits a highly available load balancer (for example, based on NGINX) to distribute license validation requests that are received in high (parallel) frequency. At the moment we assume a request every 10-15s per end user. An average cloud server should serve 20 ISVs with each having 5000 active customers (end users). On average, we predict such a server to handle approx. 10.000 parallel license requests per second. An end user could also be a machine or service representing an IoT device.

A client (end user) running protected software requires a proprietary demon. At the moment, this is a separate program (.dll) that mediates the requests to the cloud. In the future this functionality should be compiled into the protected code directly.

This demon will also initiate a point-to-point encryption between hosts as well as end-to-end encrypted channel between services which is done on basis of a proprietary implementation aligned with the main concepts of the currently emerging TLS 1.3 specification.

A dedicated keystore (HSM) will support secure management of cryptographic root keys.

License requests are distributed in a cluster of in-memory data structures (for example, REDIS) and final persistency is done in a NoSQL cluster (for example, based on MongoDB).

This core functionality is offered as a virtual machine that is running on servers with a minimum of 256 GB Ram as fast memory access is the most important technical requirement.

A key hierarchy defined by a Wibu controlled root key is responsible for granting keys to ISVs to issue license (keys). This key hierarchy is also used to enable authentication of the cloud with respect to a client.

For functionality such as managing user identities and accounts a set of REST services will be offered in combination with traditional full-stack web-frameworks (e.g. Angular or VAADIN) that will run on basis of out-of-the-box cloud services such as Amazon RDS in combination with, for example, scalable Amazon Beanstalk application servers.

3.2 Future Extensions

As part of the nationally funded "CloudProtect" project we will investigate how to use existing trust technologies in the overall scope of software protection.

As a secure element on the client side we will evaluate proven TPM functionality (for example to protect additional local encryption keys) or to serve as a random number generator. We will specifically address IoT clients running on minimal hardware such as a Raspberry 3 with an additional Optiga TPM (TPM, 2017) chip.

On both, the client as well as server side we will evaluate SGX (Intel, 2017) and TEE technologies to support isolated execution of functions.

Though we are aware of current limitations of such technologies and existing attacks (Xu et al., 2015, Brasser et al., 2017, Lee et al. 2017 and Moghimi et al. 2017) we are still convinced that we need such isolated execution environments in the long run.

First technical mitigations against known attacks against SGX technologies have been presented by the community already (Shih et al., 2017, Chen et al., 2017 and Gruss et al., 2017).

4 RELATED WORK

Software Protection has been scientifically discussed as early as (Kent, 1980), around the same time as Wibu-Systems offered the first commercial solutions as a printer port extension.

Oorschot later identified 4 approaches to software protection (Oorschot, 2003): Obfuscation via automated code-transformation; white-box cryptography; Software Tamper Resistance; and Software Diversity. Attacks on obfuscated software (Rolles et al., 2009) and the resulting improvements (Averbuch et al., 2013) are two competing disciplines and hardware supported isolated execution has been analysed extensively (Suh et al., 2007, Costan et al., 2016, Koeberl et al, 2014 and Strackx et al. 2010).

On the commercial side, there are vendors that already offer cloud-based license management (Flexera, 2018). Prominent services such as STEAM (Valve, 2018) also do, for example, offer the APIs which application developers use to enforce such access control checks. However, in both cases this is not true software protection but rather an access-control check based on a purchased license. The Steam Bind service does in fact offer cryptographic protection but has been reported to be broken (Steamless, 2016).

5 CONCLUSIONS

In this paper we shared some of our experiences in the development of an early proof of concept for a cloud-based software protection and licensing service.

We discussed traditional approaches to software protection and licensing, defined some high-level requirements for a cloud-based service, presented an architecture as well as touched on some commercial considerations of how to get this service into production and generate revenue.
This will now be further validated and extended in the context of the "CloudProtect" project funded by the German Ministry of Education & Research (BMBF) where we will provide a fully implemented proof-of-concept including trusted computing technologies as well as an analysis of the commercial dimensions.

While we cannot share too many of the technical details at this stage – mainly due to the fact that we are still in the evaluation phase of the technologies we will use for realizing, for example, the load balancer or persistence - we hope to have provided some useful insights into the applied usage of cryptography for software protection in industrial settings.

REFERENCES

Averbuch, A., Kiperberg, M., Zaidenberg, N. 2013. Truly-Protect: An Efficient VM-Based Software Protection. IEEE Systems Journal.

Brasser, F., Müller, U., Dmitrienko, A., Kostiainen, K., Capkun, S., and Sadeghi, A.-R. 2017. Software Grand

Exposure: SGX Cache Attacks Are Practical. *11th USENIX Workshop on Offensive Technologies (WOOT)*

Chen, S. Zhang, X., Reiter, M. K., Zhang, Y. 2017. Detecting privileged side-channel attacks in shielded execution with Déjá Vu. *AsiaCCS*

Collberg, C., Thomborson, C., Low D., 1997. A Taxonomy of Obfuscating Transformations. Dept. Computer Science, University of Auckland

Costan, V., Lebedev, I., Devadas, S. 2016. Sanctum: Minimal Hardware Extensions for Strong Software Isolation. *25th USENIX Security Symposium (USENIX Security)*

Flexera, 2018. https://www.flexera.com/products/software-license-optimization/flexnet-manager-cloud-infrastrucure.html

Gruss, D., Lettner, J., Schuster, F., Ohrimenko, O., Haller, I., Costa, Strong, M. 2017. Efficient Cache Side-Channel Protection using Hardware Transactional Memory. *26th USENIX Security Symposium (USENIX Security)*

Kent, S. 1980 Protecting externally supplied software in small computers. Massachusetts Institute of Technology, Cambridge, MA, USA

Koeberl, P., Schulz, S., Sadeghi, A.-R. Varadharajan, V. 2014. TrustLite: A security architecture for tiny embedded devices. *EuroSys*.

Moghimi, A., Irazoqui, G., Eisenbarth, T. 2017. CacheZoom: How SGX amplifies the power of cache attacks. Technical report, arXiv: 1703.06986

Oorschot, P. 2003 Revisiting Software Protection. *6th International Conference on Information Security*.

Rolles, R. 2009 Unpacking Virtualization Obfuscators. *3rd USENIX Workshop on Offensive Technologies (WOOT)*.

Steamless 2016 https://gitlab.com/atom0s/Steamless

Lee, S., Shih, M.-W., Gera, P., Kim, T., Kim, H., and Peinado, M. 2017. Inferring fine-grained control flow inside SGX enclaves with branch shadowing. *USENIX Security*.

Strackx, R., Piessens, F., Preneel, B. 2010 Efficient isolation of trusted subsystems in embedded systems. *Security and Privacy in Communication Networks*.

Suh, G. E., O'Donnell, C. W., Devadas, S. 2007 Aegis: A Single-Chip Secure Processor. *IEEE Design & Test of Computers*.

Intel 2017. https://software.intel.com/en-us/sgx

Shih, M.-W., Lee, S., Kim, T., Peinado, M. 2017 T-SGX: Eradicating controlled-channel attacks against enclave programs. *Network and Distributed System Security Symposium (NDSS)*.

TPM, 2017. https://trustedcomputinggroup.org/

Valve, 2018 https://developer.valvesoftware.com/wiki/Steam_Web_API

WIBU, 2018. http://www.wibu.com/de/digital-content-protection.html

Xu, Y., Cui, W., Peinado, M. 2015 Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems. *IEEE Symposium on Security and Privacy (S&P)*.