Towards Secure Dynamic Product Lines in the Cloud

Sebastian Krieter
Jacob Krüger
{skrieter,jkrueger}@hs-harz.de
Harz University of Applied Sciences
Wernigerode, Germany
Otto-von-Guericke-University
Magdeburg, Germany

Nico Weichbrodt
Vasily A. Sartakov
Rüdiger Kapitza
{weichbr,sartakov,rrkapitz}
@ibr.cs.tu-bs.de
TU Braunschweig
Braunschweig, Germany

Thomas Leich
tleich@hs-harz.de
Harz University of Applied Sciences
Wernigerode, Germany
METOP GmbH
Magdeburg, Germany

ABSTRACT

Cloud-based technologies play an increasing role in software engineering because of their scalability, availability, and cost efficiency. However, due to privacy issues, developers and organizations still hesitate to host applications that handle sensitive data on servers of external cloud providers. Modern hardware extensions, such as Intel's Software Guard Extensions (SGX), are an attempt to provide confidentiality and integrity for applications running on external hardware. Still, enabling SGX in cloud systems poses new challenges considering scalability and flexibility. In this paper, we propose an approach to address these issues by employing concepts from the domain of Dynamic Software Product Lines (DSPLs). We aim to enable applications running on SGX-based cloud systems to be securely reconfigurable and extendable during runtime. In particular, we describe properties that such an approach should fulfill and discuss corresponding challenges.

KEYWORDS

Cloud computing, SGX, Software product line, Security

ACM Reference Format:
Sebastian Krieter, Jacob Krüger, Nico Weichbrodt, Vasily A. Sartakov, Rüdiger Kapitza, and Thomas Leich. 2018. Towards Secure Dynamic Product Lines in the Cloud. In *ICSE-NIER'18: 40th International Conference on Software Engineering: New Ideas and Emerging Results Track, May 27-June 3 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3183399.3183425

1 MOTIVATION

Modern systems increasingly comprise sets of resources, including hardware and software, that are distributed among different locations. In addition, these resources are often managed by third-party providers and shared to organizations that can utilize them on-demand to scale them to their current workload. This concept is often referred to as *cloud computing* [2, 14] and facilitates a cost-efficient infrastructure for all kinds of services. However, hosting own data at an external cloud provider induces several security issues, as organizations release their potentially sensitive data into environments they do not control. Especially with more and more privacy-critical domains, reaching from medical monitoring to communications and storing of business data [5], applying cloud-computing, critical questions arise [2, 22], for example:

- Where is the data stored?
- Who has access to data and resources?
- How to ensure confidentiality and integrity of data?

To address these security issues, modern hardware technologies, such as Intel's SGX [6, 13], seem suitable. SGX aims to protect data and applications even from entities with physical access to the main memory, such as cloud providers. For this, the CPU extensions reserve and protect parts of the main memory, called *enclaves*, which can contain both, application code and regular data. Thus, even while executing application code or performing computations on others' data, the confidentiality and integrity of everything inside an enclave is ensured by SGX.

Enclaves are located in their own integrity-checked and encrypted area, called Enclave Page Cache (EPC), which has a size of 128 MiB. While enclaves can be up to 64 GiB in size, they may not fit into the EPC at once, due to complex code requirements and the memory usage of the data to be processed. Although SGX supports swapping pages between EPC and main memory, this comes with a substantial performance overhead [3] and, thus, small enclaves are preferable. This issue hampers the efficient usage of SGX for secure cloud computing, as certain tasks, such as dynamic scaling of resources and redistribution of services are difficult to achieve. Consequently, novel approaches are required to properly enable SGX in the context of such distributed and shared systems.

We propose a combination of an SGX-enabled cloud and DSPLs in an attempt to solve these issues. In addition, besides scaling in the cloud, we offer scalability on another level: DSPLs allow to modify the behavior of an application during runtime by enabling or disabling parts of its functionality [9, 10]. Using techniques for DSPLs, we aim to overcome SGX's spatial limitation by only including parts of an application that are currently needed for a given task. Furthermore, we aim to increase the efficiency of applications running in an SGX-enabled cloud by avoiding rebuilding the enclave whenever an application must be modified during runtime.

In particular, our goal is to transform any regular application into a DSPL that can run inside an enclave and is able to reconfigure itself in order to execute user-specified tasks. To this end, we require (1) a method that supports developers in partitioning their application

ACM ISBN 978-1-4503-5662-6/18/05…$15.00
https://doi.org/10.1145/3183399.3183425

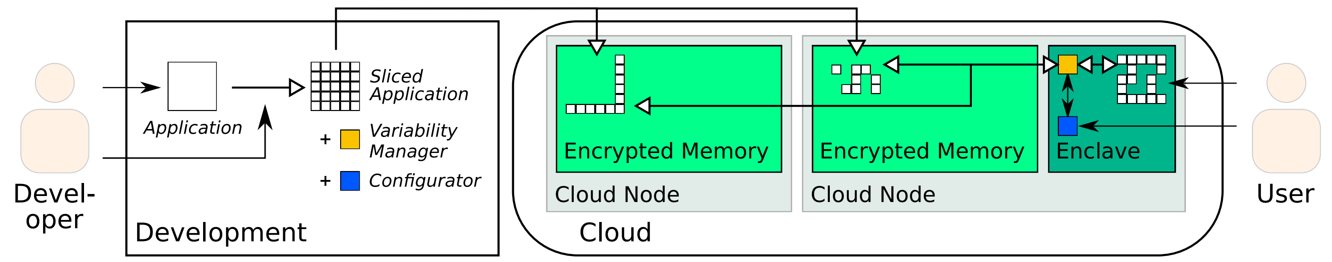

Figure 1: Sketch of the envisioned approach for scalable SGX in distributed systems.

into variable parts, (2) an adaptation mechanism that is compatible with the SGX environment, and (3) a configurator that is able to find a suitable configuration for a given task.

Compared to current techniques for SGX [4, 12, 23], in addition to providing protection for all code and data against unauthorized access, such an approach enables:

- Flexible control over application functionality at runtime.
- Scalability to include only necessary code into the enclave.
- Dynamic loading of third-party services at runtime.

In summary, we envision to apply DSPLs to SGX-enabled clouds in order to achieve secure, yet scalable applications. With this paper, we sketch our envisioned approach, discuss each step, and address the corresponding challenges.

2 ENVISIONED PROPERTIES

Our overarching goal is to improve the security of applications and data, while we can still utilize the scalability and availability of cloud systems. For this purpose, we aim to achieve the following properties:

P_1 *Protecting the entire code base from unauthorized access.*
We aim to prevent any kind of unauthorized access or manipulation of application code and data. To this end, we either keep currently needed code and data inside an enclave or store it securely in other parts of the memory (cf. Section 3.1).

P_2 *Running sliced applications within the enclave.*
If an application does not completely fit into the SGX-secured memory, we partition its functionality and only include those parts that are currently needed (cf. Section 3.1).

P_3 *Enabling dynamic loading of variable application parts.*
We require a mechanism to exchange parts of an application at runtime to avoid rebuilding the enclave and restarting the application. This way, we aim to prevent downtime and increase flexibility of the application (cf. Section 3.2.1).

P_4 *Supporting self-adaptive reconfiguring of applications.*
Instead of configuring the application manually, we want it to reconfigure itself to adapt to a given task. Thus, the application should be able to decide which features are required for the current task (cf. Section 3.2.2).

P_5 *Utilizing the scalability and availability of cloud computing.*
Our envisioned approach allows for secure and scalable execution of applications on a single cloud node. However, in the context of distributed computing, performance is scaled by utilizing several nodes, which we have to enable for our concept. Thus, we have to ensure that an application can be securely and consistently configured across different nodes (cf. Section 3.2.3).

P_6 *Including and securing third-party services.*
A central concept of cloud computing is to provide third-party services to its users. To reuse such services, we have to provide mechanisms to integrate them dynamically into the application by extending the set of variable parts (cf. Section 3.2.4).

By fulfilling these properties, we aim to increase the security, flexibility, and availability of applications that are running in an SGX-enabled cloud. In the following section, we sketch our approach by considering the development of a secure DSPL, its reconfiguration during runtime, and the utilization of distributed resources.

3 SECURE CLOUD PRODUCT LINES

In our approach, we aim to combine three different concepts: Cloud computing, DSPLs, and SGX. Cloud computing ensures scalability and availability of applications, due to access to distributed resources. Combining this with SGX and DSPLs aims to guarantee security and flexibility of applications on a single cloud node. A combination of DSPLs and an SGX-enabled cloud, means we have to develop applications that can run and reconfigure themselves inside an enclave. Consequently, we propose a two-phase approach: First, an initial transformation of an application into a DSPL. Second, the execution and adaptation of the application within the SGX-enabled cloud node. We depict our envisioned approach in Figure 1 and discuss the displayed steps and components in this section, emphasizing the arising challenges of combining these concepts.

3.1 Developing an Application

In order to transform an application into a DSPL, we need to apply three modifications: First, we need to define variability within the source code. In other words, we have to slice or partition the application into separated parts, called *features*, of which each comprises a single functionality. This can be done with different software-product-line techniques, such as annotating or modularizing the source code [1, 8]. Second, we have to define interdependencies of these variable parts within a variability model [19]. Each configuration of the application must be in accordance with this variability model to guarantee correct and secure behavior. Third, we need to specify an adaption mechanism that implements the runtime variability of the DSPL [18]. Such an adaption mechanism is responsible for loading and binding features of the software product line during runtime and discarding features that are no longer needed.

This mechanism is a crucial part as it must be interleaved with the SGX functionalities, including runtime decryption and integrity checking. In Figure 1, we display the original application on the left side. During development, it is sliced into features that can be dynamically configured and included into the enclave.

Utilizing the aforementioned modifications, the DSPL still needs to be reconfigured externally, for instance by a developer. To cope with changing application tasks, resources, and services as they appear in the cloud, we also require self-adaptation of the software product line to react to such changes. For this case, we need to provide an additional configurator unit, which is able to find a valid configuration that corresponds to the changed context. In particular, the configuration has to ensure the quality of the reconfiguration to avoid any security breaches, which is a problem in sensitive and safety-critical systems [11].

We consider the first two steps, identifying meaningful features and specifying their interdependencies, as a particularly challenging task. Since a feature should comprise a single, distinct functionality, in most cases, it requires the expertise of a developer and, thus, is hard to fully automatize. Thus, one of the main challenges is to define heuristics that support the developer in identifying suitable features. Possible methods are, amongst others, static dependency analyses or analyzing natural language documents, such as, source code documentation or commit messages.

Challenge:
How can we identify meaningful features within an unsliced application?

3.2 Running in the Cloud

Once we have build a DSPL, we have to run it on an SGX-enabled cloud node and ensure its scalability. We depict this part of our approach on the right side of Figure 1.

3.2.1 Variability in the Enclave. While there exist approaches [4, 12, 23] to run parts of and even full applications in an SGX-enabled cloud, these are not dynamic. Thus, in case the application must be modified, it is necessary to rebuild the enclave and manually specify changes. As we aim to include features into the enclave on-demand, we need to develop an adaption mechanism, which we call *variability manager* (marked in orange in Figure 1). While a DSPL allows for dynamic changes, we need a variability manager to control this runtime variability within the SGX enclave. The variability manager performs communications between the enclave and the remaining memory. It is responsible to load features, including decryption and integrity checking, and removing features from the enclave.

Challenge:
How can we efficiently load single features of a DSPL into an enclave?

3.2.2 Self-Adaption at Runtime. Due to the high availability that cloud services promise and that are necessary in many modern environments, such as cyber-physical systems or health monitoring, the current practice of configuration at design-time seems unsuitable [11, 17]. Any halt of the application to apply changes stops all computations that are performed. Thus, we aim to enable each DSPL to work within the enclave and reconfigure itself at runtime to fulfill its current task.

While the variability manager can load features from outside into the enclave, the actual decision, which features are needed at which point in time, must be handled by another component, the *configurator* (marked blue in Figure 1). In order to apply on-demand changes, the configurator needs to analyze the task at-hand and identify which features need to be included in the enclave to fulfill a task. This way, we aim to ensure scalability by ensuring that only necessary parts are within the enclave. For this purpose, it is necessary to provide feature and task specifications to the configurator. By matching these specifications, the configurator is empowered to decide which features are required and which can be discarded.

Challenges:
What are meaningful heuristics for self-adaptation?
Especially, at which point should features be removed from the enclave?

3.2.3 Utilizing the Cloud. We argue that the described approach ensures secure and scalable program execution on a single cloud node. However, using only one node is not the intended scenario of cloud computing, which scales applications among its shared and distributed resources. Thus, we also need to consider that our approach may has to be moved to another node, for example if the current one crashes, or that the application uses parallel computation on multiple nodes, for example as in map-reduce [7].

While most cloud providers guarantee a high uptime in their service level agreements, a temporary downtime is still possible [17]. In addition, the running services are partly redistributed to scale to changing demands. Thus, our approach must be able to switch the application and enclave to another cloud node. At this point, it is necessary to ensure consistent and secure storing of the application's current configuration to prevent any security breaches.

Distributed computation with SGX just started to gain attention by researchers [16, 20]. Considering our approach, we need to develop techniques that can ensure secure and efficient distribution of code and data among enclaves on different cloud nodes. Here, problems arise due to the variable configuration that is used by the DSPL. In contrast to the self-adaptive behavior that we considered before, we now require one application to coordinate the distributed computation. More precisely, this application has to ensure that all others use the same configuration, provide the input data to them, and merge the results. Thus, we have to partly discard the self-adaptive behavior of our approach while different applications work on the same task.

Challenges:
How can we efficiently store and transfer the internal configuration of running DSPL among different nodes?
How can we efficiently apply distributed computation in different DSPLs among different nodes?

3.2.4 Integrating Cloud Services. Another aspect of cloud computing is to provide different services to use them in own applications. Thus, instead of developing everything from scratch, existing implementations can be reused. However, as we use DSPL to enable dynamic loading, further challenges arise.

As first question arises, how we can include third-party services into an application. Here, problems are connected to the size of a service and its inclusion into the DSPL. While an advantage of DSPLs is their adaptability to new functionality at runtime, the specification of a service may be unknown and has to be dynamically included. Furthermore, the service may require slicing in order to reduce its size. Both may be challenging if the service's developer does not provide a partitioned version, which seems unlikely. Thus, we need to apply approaches for program slicing and provide suitable interfaces to include required parts into a DSPL.

When we can include a service or its parts into the enclave, the next question is, how we can ensure security of our data. SGX ensures the security of everything that is within its enclave. However, if we include corrupted or manipulated code and data, we still face a security problem. Mainly, we see the issue that services may inject code to monitor data within the enclave. As we cannot verify their source code, but still need the service, additional security means seem necessary.

Challenges:
How can we efficiently include third-party services into our DSPL?
How can we efficiently check the integrity of services and their parts?

4 RELATED WORK

In previous research, systems that load full applications into an enclave have emerged. SCONE [3] and Haven [4] allow running unmodified Linux and Windows applications inside SGX, respectively. Graphene-SGX [23] furthermore enables dynamic linking for enclave applications. The cost of system calls in enclaves has been decreased by Panoply [21] and Eleos [15]. However, those projects always load the full application and have a loader or library OS inside the enclave. Eleos also enables user-level paging of enclave data but not code, a feature we aim to provide.

The approach closest to our own is Glamdring [12], which aims to slice applications into security-sensitive and non-security-sensitive features. Glamdring (1) requires annotations for security-sensitive data that has to be protected and (2) is application-dependent. Contrarily, we aim to secure all computation, which increases the effort but allows to include any application.

Considering these characteristics, we have to considerably extend existing approaches and introduce dynamic SGX slicing.

5 CONCLUSION AND FUTURE WORK

Cloud-computing has become a central part of software engineering to ensure scalability and availability of services. However, confidentially and integrity of the outsourced data plays an equally important role. For a way to combine the advantages of both worlds, we propose an approach utilizing DSPLs in an SGX-enabled cloud and identified several challenges that need to be addressed in order to achieve this goal. In our opinion, investigating these challenges and developing corresponding solutions is the key to combine three promising concepts – enabling secure, scalable, and flexible computing in the cloud. During our future research we, will implement our approach and compare it to others.

ACKNOWLEDGMENTS

This research is supported by DFG grants LE 3382/3-1, KA 3171/6-1, and LE 3382/2-1, and Volkswagen Financial Services AG.

REFERENCES

[1] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. 2013. *Feature-Oriented Software Product Lines.* Springer.

[2] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. 2010. A View of Cloud Computing. *Communications of the ACM* 53, 4 (2010), 50–58.

[3] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O'Keeffe, Mark L Stillwell, David Goltzsche, David Eyers, Rüdiger Kapitza, Peter Pietzuch, and Christof Fetzer. 2016. SCONE: Secure Linux Containers with Intel SGX. In *USENIX Conference on Operating Systems Design and Implementation (OSDI).* USENIX Association, 689–703.

[4] Andrew Baumann, Marcus Peinado, and Galen Hunt. 2015. Shielding Applications from an Untrusted Cloud with Haven. *ACM Transactions on Computer Systems* 33, 3 (2015), 8:1–8:26.

[5] Nathalie Brender and Iliya Markov. 2013. Risk Perception and Risk Management in Cloud Computing: Results from a Case Study of Swiss Companies. *International Journal of Information Management* 33, 5 (2013), 726–733.

[6] Victor Costan and Srinivas Devadas. 2016. *Intel SGX Explained.* Technical Report. IACR Cryptology ePrint Archive.

[7] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM* 51, 1 (2008), 107–113.

[8] Critina Gacek and Michalis Anastasopoules. 2001. Implementing Product Line Variabilities. *ACM SIGSOFT Software Engineering Notes* 26, 3 (2001), 109–117.

[9] Svein Hallsteinsen, Mike Hinchey, Sooyong Park, and Klaus Schmid. 2008. Dynamic Software Product Lines. *Computer* 41, 4 (2008), 93–95.

[10] Mike Hinchey, Sooyong Park, and Klaus Schmid. 2012. Building Dynamic Software Product Lines. *Computer* 45, 10 (2012), 22–26.

[11] Jacob Krüger, Sebastian Nielebock, Sebastian Krieter, Christian Diedrich, Thomas Leich, Gunter Saake, Sebastian Zug, and Frank Ortmeier. 2017. Beyond Software Product Lines: Variability Modeling in Cyber-Physical Systems. In *International Systems and Software Product Line Conference (SPLC).* ACM, 237–241.

[12] Joshua Lind, Christian Priebe, Divya Muthukumaran, Dan O'Keeffe, Pierre-Louis Aublin, Florian Kelbert, Tobias Reiher, David Goltzsche, David Eyers, Rüdiger Kapitza, Christof Fetzer, and Peter Pietzuch. 2017. Glamdring: Automatic Application Partitioning for Intel SGX. In *USENIX Annual Technical Conference (ATC).* USENIX Association, 285–298.

[13] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. 2016. Intel&Reg; Software Guard Extensions (Intel&Reg; SGX) Support for Dynamic Memory Management Inside an Enclave. In *Hardware and Architectural Support for Security and Privacy (HASP).* ACM, 10:1–10:9.

[14] Peter Mell and Tim Grance. 2011. *The NIST Definition of Cloud Computing.* Technical Report. National Institute of Standards and Technology.

[15] Meni Orenbach, Pavel Lifshits, Marina Minkin, and Mark Silberstein. 2017. Eleos: ExitLess OS Services for SGX Enclaves. In *European Conference on Computer Systems (EuroSys).* ACM, 238–253.

[16] Rafael Pires, Daniel Gavril, Pascal Felber, Emanuel Onica, and Marcelo Pasin. 2017. A Lightweight MapReduce Framework for Secure Processing with SGX. In *International Symposium on Cluster, Cloud and Grid Computing (CCGrid).* IEEE, 1100–1107.

[17] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. 2009. A Taxonomy and Survey of Cloud Computing Systems. In *International Joint Conference on INC, IMS and IDC (NCM).* IEEE, 44–51.

[18] Marko Rosenmüller, Norbert Siegmund, Mario Pukall, and Sven Apel. 2011. Tailoring Dynamic Software Product Lines. In *International Conference on Generative Programming and Component Engineering (GPCE).* ACM, 3–12.

[19] Ina Schaefer, Rick Rabiser, Dave Clarke, Lorenzo Bettini, David Benavides, Goetz Botterweck, Animesh Pathak, Salvador Trujillo, and Karina Villela. 2012. Software Diversity: State of the Art and Perspectives. *International Journal on Software Tools for Technology Transfer* 14, 5 (2012), 477–495.

[20] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. 2015. VC3: Trustworthy Data Analytics in the Cloud Using SGX. In *Symposium on Security and Privacy (SP).* IEEE, 38–54.

[21] Shweta Shinde, DL Tien, Shruti Tople, and Prateek Saxena. 2017. Panoply: Low-TCB Linux Applications with SGX Enclaves. In *Network and Distributed System Security Symposium (NDSS).* The Internet Society, 12.

[22] Subashini Subashini and Veeraruna Kavitha. 2011. A Survey on Security Issues in Service Delivery Models of Cloud Computing. *Journal of Network and Computer Applications* 34, 1 (2011), 1–11.

[23] Chia-Che Tsai, Donald E Porter, and Mona Vij. 2017. Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX. In *USENIX Annual Technical Conference (ATC).* USENIX Association, 645–658.