Securing Orchestrated Containers with BSI Module SYS.1.6

Christoph Haar and Erik Buchmann
Hochschule für Telekommunikation Leipzig, Gustav-Freytag-Str. 43-45, 04277 Leipzig, Germany

Keywords: IT-Grundschutz, IT-Security, Container Virtualization, Kubernetes, Orchestration.

Abstract: Orchestrated container virtualization, such as Docker/Kubernetes, is an attractive option to transfer complex IT ecosystems into the cloud. However, this is associated with new challenges for IT security. A prominent option to secure IT infrastructures is to use security guidelines from agencies, such as Germany's Federal Office for Information Security. In this work, we analyze the module ”SYS.1.6 Container” from this agency. We want to find out how suitable this module is to secure a typical Kubernetes scenario. Our scenario is a classical 3-tier architecture with front end, business logic and database-back end. We show that with orchestration, the protection needs for the entire Kubernetes cluster in terms of confidentiality, integrity and availability automatically become ”high” as soon as a sensitive data object is processed or stored in any container. Our analysis has shown that the SYS.1.6 module is generally suitable. However, we have identified three additional threats. Two of them could be exploited automatically, as soon as a respective vulnerability appears.

1 INTRODUCTION

In 2019, a quarter of all company data was stored in a cloud (Joseph McKendrick, 2018). In this context, container virtualization is frequently used. This allows planning, operating and maintaining complex applications, such as database management systems (DBMS) in an agile and cost-efficient way, which is also compatible with DevOps approaches. A prominent container virtualization is the open source project Docker (Docker Inc., 2020b). Docker containers allow managing applications, such as DBMS as separate building blocks of a large IT Infrastructure. For example, it is possible to place a preconfigured operating system image together with a preconfigured image of a database management system in a container. This container can be evaluated it in a test environment, transferred to a productive system and duplicated on several host computers if the demand for resources increases.

In this paper, we use DBMS as our running example. The Docker repository (Docker Inc., 2020a) currently lists 6,238 container images with the keyword ”database”. This includes relational DBMS, such as Oracle, Postgres or SQL Server, and special systems, such as Couchbase, MongoDB or MariaDB. In addition to the operating system image and one or more application images, a container can also contain sensitive company data or personal information. To manage different container instances across multiple hosts, an orchestration like Kubernetes (The Kubernetes Authors, 2020) is frequently used. The orchestration monitors and assigns resources, such as storage or computing power, controls network access and isolates the container instances against each other. It also allocates containers to virtual machines and virtual machines to physical hosts in a data center.

Due to the orchestration, the security measures implemented in a DBMS lose their importance as the first line of defense between sensitive data, users and public networks. The DBMS cannot control the resources of its tenants, if the orchestration can withdraw resources from containers (Kanchanadevi et al., 2019). The orchestration also controls the data exchange between containers, i.e., it ignores the settings stored in the DBMS (Gao et al., 2017). Each container imports its own DBMS instance via an image. Typically, each DBMS instance is only responsible for one database, and uses only one administrator and one user account. Therefore, the isolation between the DBMS clients is forwarded to the container isolation (Mardan and Kono, 2020). Since the container instances only exist in the main memory, persistent data must be stored on another host. This means that the protection needs for the managed data must be transferred from the DBMS to the container virtualization and orchestration, and consider multiple hosts in a complex IT ecosystem.

Haar, C. and Buchmann, E.
Securing Orchestrated Containers with BSI Module SYS.1.6.
DOI: 10.5220/0010340406760683
In *Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP 2021)*, pages 676-683
ISBN: 978-989-758-491-6

Standardized compilations of security guidelines allow to systematically develop security concepts. Examples are the IT-Grundschutz (BSI, 2011) from the Federal Office for Information Security (BSI), the ISO 27001 certification (BSI, 2014), or the Information Security Practice Guides (NIST, 2020) from the National Institute of Standards and Technology (NIST). In this work, we focus on the IT-Grundschutz approach from BSI. Currently, the BSI is developing the module ”SYS.1.6 Container” (BSI, 2020). We have already shown that SYS.1.6 is applicable to secure containers, which are executed on a single host (Haar and Buchmann, 2019). In this work, we use a typical database scenario to analyze the suitability of the 2020 draft of SYS.1.6 to secure an orchestrated container virtualization.

We have developed a protection concept according to IT-Grundschutz for a typical Kubernetes scenario (Vaughan-Nichols S., 2020). This scenario consists of a DBMS with customer data, a business logic including payment system, and a web application as a front end. We model the information domain for our Kubernetes system according to BSI standard 200-2 (BSI, 2017a). Then we determine the protection needs and analyze the elementary threats described in the BSI module SYS.1.6 Container. Because some data objects in the database in the information domain require the protection need ”high”, we will do a risk analysis according to BSI standard 200-3 (BSI, 2017b) in a second step to identify and evaluate additional threats to our Kubernetes System.

We have observed that in an orchestrated container virtualization system, a single data object with the protection need ”high” for confidentiality, integrity or availability ensures that the entire system must be assigned with this protection need. This is, because the orchestration decides at run-time which instances of a container are running and where. Our analysis has shown that SYS.1.6 is suitable for securing such a scenario. However, we have found three additional threats that are not considered in SYS.1.6. Two threats could be used to implement an automated exploit, as soon as an attacker finds a corresponding vulnerability. Note that SYS.1.6 from the BSI considers the same set of security risks as the ”Application Container Security Guide” (NIST, 2017) from NIST. Thus, our results can be transferred to the Information Security Practice Guides.

Structure of the Work. Section 2 describes Docker, Kubernetes and IT-Grundschutz. In Sections 3 and 4 we perform a risk analysis and compare our findings with those from BSI. Section 5 concludes.

2 RELATED WORK

In this section, we explain the standard protection and risk analysis according to BSI, the module SYS.1.6 (BSI, 2020) and the basics of Docker and Kubernetes.

2.1 BSI Standard-protection

The BSI standard 200-2 (BSI, 2017a) defines six steps to secure a typical IT system. We use these steps as our research method.

1. The **scope** must be defined first. The scope is the information domain to be protected.
2. In the **structure analysis**, the processes, applications, IT-Systems, infrastructures, etc. within the scope are defined as **target objects**.
3. The third step **defines protection needs** for the business processes, the information processed and the information technology used.
4. In the **modeling** step, the **modules** of the IT-Grundschutz-Kompendium (BSI, 2019) are used to identify security measures for the target objects, depending on the protection needs.
5. The **IT-Grundschutz-Check** finds out if the implemented security measures are sufficient to fulfill the protection needs.
6. A **risk analysis** must be implemented if a target object has protection needs above normal, if if no BSI module exists for a target object, or if a target object is operated in an unusual way.

The risk analysis identifies **additional threats** that are not considered in the modules. The BSI standard 200-3 (BSI, 2017b) provides a set of questions that help to perform such a risk analysis. These questions should be answered by experts, employees, administrators and users for each target object:

- Which ”force majeure” threats are relevant?
- Are there organizational deficiencies that have an impact on information security?
- Can the safety be compromised by human errors?
- Do technical failures result in security problems?
- Which threats can arise from external attacks?
- Is it possible for employees to willfully impair the operation of the target object?
- Is it possible that objects outside of the information system cause a risk?
- What information is provided by the manufacturer’s documentation and third parties?

Table 1: Standard-Requirement SYS.1.6.A26.

Requirement	Description
SYS.1.6.A26	*Service accounts in containers* The accounts used by the processes inside a container application SHOULD not have permissions on the host that executes the container environment. If this is necessary, these permissions SHOULD be restricted on the data that is absolutely necessary.

2.2 Module SYS.1.6: Container and Application Container Security

In the BSI-Grundschutz-Compendium (BSI, 2019), each module secures a specific target object. A module contains a description of the target object, the objectives of the protection, cross-references to other objects. The main part of each module is a description of specific threats for the target object, together with a set of requirements to avoid these threats. Finally, each module lists standard security measures.

The requirements are divided into (1) **basic requirements** that must be implemented as quickly as possible, (2) **standard requirements** that should be implemented to achieve basic protection, and (3) requirements that must be implemented when there is an **increased need** for protection. With pre-defined terms like SHOULD or MUST, the module points out the importance of the requirements. Table 1 shows an example of the requirement A26 of the SYS.1.6 module. A26 belongs to ”standard requirements”.

To secure an orchestrated container virtualization with Kubernetes, the BSI has published a draft of module SYS.1.6 ”Container” (BSI, 2020). To provide an overview, we briefly list the basic requirements (Table 2), standard requirements (Table 3) and the requirements for increased protection (Table 4).

Table 2: Basic requirements.

ID	Basic requirements
SYS.1.6.A1	Container use
SYS.1.6.A2	Separation of container apps
SYS.1.6.A3	Administration and orchestration
SYS.1.6.A4	Hardening of the host system
SYS.1.6.A5	Separation of containers
SYS.1.6.A6	Trustworthy images
SYS.1.6.A7	Hardening software in a container
SYS.1.6.A8	Persistence of logging data
SYS.1.6.A9	Persistence of user data
SYS.1.6.A10	Storing login information
SYS.1.6.A11	Administrative remote access

Note that the english translation of the module ”SYS.1.6 Container” is not available at the moment, because the module has not been finalized yet. However, the module borrows from the ”Application Container Security Guide” of the National Institute of Standards and Technology (NIST) (NIST, 2017). Because of this, both approaches handle the same kinds of risks. In particular, the basic- and standard requirements and requirements for a higher protection needs from the BSI module correspond to the countermeasures in section four of the Application Container Security Guide. Thus, our findings can be transferred to the Application Container Security Guide.

Table 3: Standard requirements.

ID	Standard requirements
SYS.1.6.A11	Policy for operation and images
SYS.1.6.A12	One service per container
SYS.1.6.A13	Image and conf. approval
SYS.1.6.A14	Updating containers
SYS.1.6.A15	Immutability of the container
SYS.1.6.A16	Limiting container resources
SYS.1.6.A17	Mass storage for containers
SYS.1.6.A18	Securing admin. networks
SYS.1.6.A19	Separate in- and output systems
SYS.1.6 A20	Backup of configuration data
SYS.1.6 A21	Unprivileged containers
SYS.1.6 A22	Securing auxiliary processes
SYS.1.6 A23	Administrative remote access
SYS.1.6 A24	Identity and auth. management
SYS.1.6 A25	Service accounts for containers
SYS.1.6 A26	Accounts in containers
SYS.1.6 A27	Monitoring containers
SYS.1.6 A28	Securing the image registry

Table 4: Requirements for high protection needs

ID	Requirements for high protection
SYS.1.6.A29	Automated container auditing
SYS.1.6.A30	Private image repositories
SYS.1.6.A31	Strict access policies
SYS.1.6.A32	Host-based intrusion detection
SYS.1.6.A33	Container micro-segmentation
SYS.1.6.A34	Container high availability
SYS.1.6.A35	Encrypted data storage
SYS.1.6.A36	Encrypted communication

2.3 Orchestration and Kubernetes

Orchestration (TechnologyAdvice, 2020) allows to efficiently control complex IT-Service landscapes in the company. The most commonly used platform for orchestrating containers is Kubernetes (Vaughan-Nichols S., 2020). Other platforms for orchestrating containers use similar concepts, e.,g., Apache Mesos (Foundation, 2020), AWS Fargate (Services, 2020) or Cloudify (Ltd., 2006). See (A., 2019) for a detailed description.

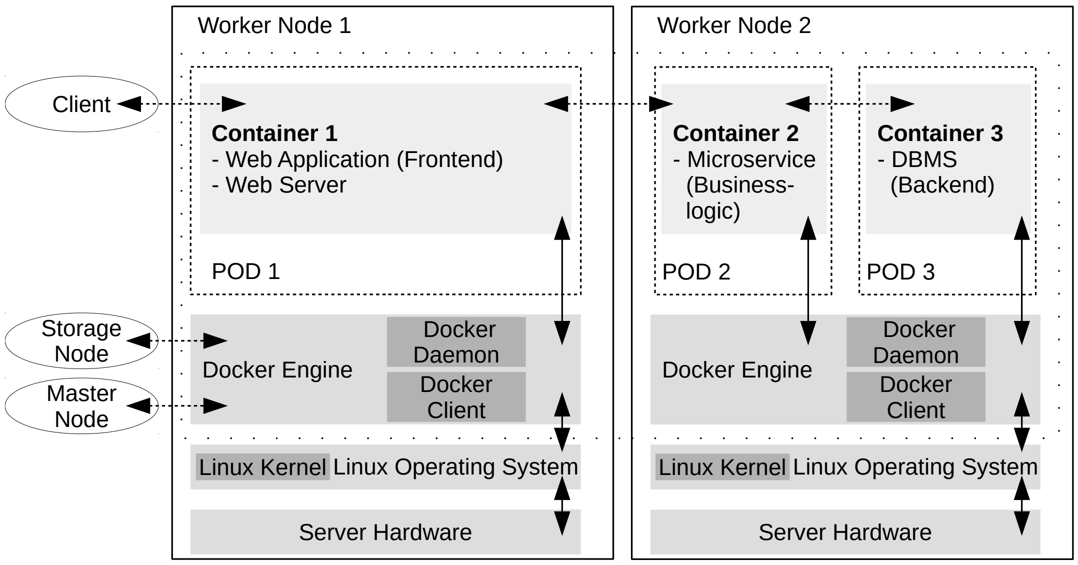

Figure 1: Example-System Web-Shop.

Kubernetes is an open source platform developed by Google, whose primary task is to monitor and control the resource utilization of individual containers on the respective hosts. Kubernetes can start, shut down or move container instances over multiple hosts depending on the workload of the system and free resources of the hosts. Furthermore, Kubernetes can monitor the state of containers, it allows to configure authorizations or migrate containers (Sayfan, 2018).

Kubernetes consists of several components. A **worker node** is a physical host or a virtual machine on which containers are operated. The**master node** is the host that is responsible for controlling and monitoring the resources on the worker nodes. The **etcd** (The etcd authors, 2020) on the master node is a key-value store for the configuration data of all containers and services. The **Kubectl** tool transmits commands for administration to the master node (Oliveira et al., 2016). A **kubelet** is an execution environment for PODs on a worker node. **PODs** are a set of one or more container instances that are managed together (Sayfan, 2018) and use resources of the same host. The system of worker and master nodes, as well as other storage and network resources is a **Kubernetes cluster**. Each worker node has a **proxy** that forwards TCP/UDP packets to/from PODs. The **REST API** server enables the user to establish communication between master and worker nodes (Sayfan, 2018). The **scheduler** monitors and plans individual PODs, which is communicated via the REST API server (Baier, 2017).

3 SECURING KUBERNETES

In this section, we describe a typical Kubernetes scenario. For this scenario, we develop the security needs of the BSI level ”Standard Protection”.

3.1 Information Domain

Szenario: *A retailer stores a relational database with data from a web shop. The data include customer data, orders and payment transactions. The web shop is also equipped with an Internet payment system that securely processes payment transactions through various channels via a service provider. For this purpose, the retailer has divided a classical 3-tier architecture into three Docker containers, as shown in Figure 1: The first container contains the front end. This includes a web server with a PHP-based web application. The second container contains the business logic and the payment system as a set of REST-based microservices written in Java. A Postgres DBMS with the database is housed in the third container. All containers are managed by Kubernetes. Kubernetes does not use further add-ons. Each host in the Kubernetes cluster contains a POD that can run multiple instances of the containers. The Kubernetes scheduler decides, depending on the workload, how many instances of which container are executed within which POD. Only the database is executed in a single instance, so that there is no need to synchronize multiple databases. The Kubernetes cluster runs in an on-premise environment, i.e., the retailer is responsible* *for the containers and the data-center infrastructure.*

According to BSI standard 200-2 (BSI, 2017a), the information domain must be modeled before the protection needs are determined. Table 5 contains all categories of data objects from our scenario. Table 6 contains all applications that are required to operate the web shop. The software used is summarized in table 7. Table 8 shows all host systems within the Kubernetes cluster. All hosts belong to the same data center (DC1). The master node (S0) and the worker nodes (S1, S2) controlled by it are housed on their own hosts. Instances of the containers (C1 - C3) with web applications, microservices and DBMS run on the worker nodes.

Table 5: Data of the Information Domain.

Nr.	Data Object	Description
D1	Personal Data	Data from a natural person
D2	Payload	Data from applications and service operations
D3	Account Data	Login data and authorization data
D4	Configuration Data	Data that controls the system behavior
D5	Log Data	Data on past operations and transactions

Table 6: Applications of the Information Domain.

Nr.	Descript.	Data	Software	Con
A1	Web App.	D1, D2, D3, D4, D5	App. logic in PHP	C1
A2	Web Server	D1, D2, D4, D5	Apache	C1
A3	Micro-service	D2, D3, D4, D5	REST service	C2
A4	Database	D1, D2, D3, D4, D5	Postgres database	C3

Table 7: Software of the Information Domain.

Nr.	Description	Data	IT System
SSW1	Docker Software	D4, D5	S1, S2
SSW2	Kubernetes Software	D4, D5	S0

Since we focus on the orchestration, we have summarized the hardware and the operating system under S0 to S2 ”Host system”. In the following, we assume that these objects are already protected according to IT-Grundschutz.

Table 8: Host-Systems of the Information Domain.

Nr.	Descript.	Data	Platform	Loc.
S0	Host1	D1, D2, D3, D4, D5	x86 Linux	DC1
S1	Host2	D1, D2, D3, D4, D5	x86 Linux	DC1
S2	Host3	D1, D2, D3, D4, D5	x86 Linux	DC1

3.2 Protection Needs

The aim of defining protection needs is to find out which protective measures are appropriate for the respective objects in the information system. According to the standard 200-2 (BSI, 2017a), the BSI defines three protection need categories ”normal”, ”high” and ”very high”. In the following we will define the protection needs for our Kubernetes cluster. We use the protection need categories proposed by the BSI.

We start defining the protection needs by determining the protection needs of the individual data objects (D1-D4), which are stored, processed and transferred between the containers within our Kubernetes cluster. In the next step, the data protection needs are passed on to the applications (A1-A4) that use the corresponding data, and from there to the individual worker nodes (S1 and S2) and the master node (S0). This means that the protection needs of the data stored in the database are transferred to the container with the database application and to all other containers that work on this database. The protection needs are then transferred to the Kubernetes cluster and from there to the physical hosts.

If data with different protection needs is stored in the same Kubernetes cluster, the highest of these protection need is assigned to the entire Kubernetes cluster. This results in a special case for an orchestrated container virtualization:

The individual worker nodes are all connected to each other via a central master node. This master node can access all data of the individual worker nodes. E.g. on the customer data in the database, as well as on the configuration data and the applications loaded via image. Depending on its configuration at runtime, the orchestration controls which containers are started in which POD.

For this reason, the protection needs of each individual worker node are passed on to the entire Kubernetes cluster and finally to the entire information domain. In addition, continuous availability of the Kubernetes cluster is only guaranteed if at least one worker node and the master node are operational. Therefore, only the highest protection need with regard to confidentiality, integrity and availability of the data have to be analyzed to determine the protection needs. From there, the protection needs are passed on to all containers, all PODs that manage these containers, and all worker nodes that execute the PODs.

This means for our application scenario:

- **Confidentiality.** D1 and D3 store personal data, which always have at least the protection need "high". The orchestration determines at runtime on which worker node container instances with DBMS and the other applications are executed. For this reason, confidentiality must be "high" for every worker node. The same applies to the master node, as it is controls the worker nodes. This is why the entire information system has a high need for protection with regard to confidentiality.
- **Integrity.** A central service in our scenario is the processing of customer payment transactions. Payment data (D1, D2) must not be changed without authorization. Because it is also not known in advance on which worker node the payment service is running, the protection needs of the information system with regard to integrity are "high".
- **Availability.** If one of the applications (A1-A4) is discontinued, the web shop is out of service. For this reason, the availability must be "high" for any target object in the entire information domain.

4 KUBERNETES THREATS

In the previous section, we have learned that the security demand for the entire system is "high" for integrity, confidentiality and availability. This is due to the fact that the orchestration decides at run-time on which host sensible data objects are managed. For any risk level above "normal", BSI standard 200-3 (BSI, 2017b) requires us to do a risk analysis to (i) identify and (ii) assess additional threats.

4.1 Identifying Additional Threats

We performed a risk analysis as described in Section 2.1. Our experts were part-time students and employees of the IT-Departments of Deutsche Telekom AG and T-Systems GmbH, with many years of professional experience. Our experts have identified 10 different additional threats for our scenario (see Table 9). Seven of them are identical to the threats which we have already identified for container virtualization without orchestration, i.e., we were able to confirm our previous work (Haar and Buchmann, 2019). By now, those threats are also part of the BSI module (BSI, 2020).

Table 9: Additional Threats for the Kubernetes-System.

Identified in SYS.1.6
Vulnerabilities in images
Insecure administrative access
Orchestration with insecure tools
Container breakout
Data loss due to lack of persistence
Loss of confidentiality of login information
Unauthorized modification of configuration data
Not identified in SYS.1.6
Bad planning of etcd
Compromising the nodes
Unauthorized access to the etcd

4.2 Risk Assessment

For each additional threat, the BSI standard (BSI, 2017b) requires a qualitative risk assessment to quantify the potential danger. In the following, we perform this risk assessment for the three risks the module SYS.1.6 does not consider so far.

The risk is the product of the frequency of occurrence and the extent of the damage (BSI, 2017b). We have used a categorization of occurrences and damages, which is in line with the BSI recommendation. A detailed categorization can be found in (Haar and Buchmann, 2019).

Tables 10 to 12 contain our risk assessment. Each table lists the object from the information system to which the threat relates, the threat itself, the impaired basic values, frequency of occurrence, extent of damage, risk and an intuitive description.

We already knew from preliminary work (Haar and Buchmann, 2019) that executing database systems in a container virtualization leads to new security challenges. If the DBMS is operated on its own host, the operator can directly control which database and system rights exist, when which security patches are imported, and when the DBMS is started or shut down. This becomes more difficult with container virtualization. The orchestration increases these challenges. Due to the orchestration, a large number of containers can be started, stopped, modified, reconfigured or relocated at once. Therefore, the orchestration is an attractive target for an attacker. The SYS.1.6 module offers a valuable support when protecting such a system. However, our risk analysis has shown that three kubernetes-specific threats are not included in the module. "Compromising the nodes" and "Unauthorized access to the etcd" are even suit-

Table 10: Risk: Bad Planning of the etcd.

Kubernetes-System (*Confidentiality: high, Integrity: high, Availability: high*)		
Threat: **Bad Planning of etcd**	Impaired Core Values: Availability	
Frequency of Occurrence without add. Measures: medium	**Extent of Damage** without add. Measures: considerable	**Risk** without add. Measures: high
Description: A bad planning of the key-value-store, which serves as persistent storage for all data of all clusters (etcd) can be caused by inattentive or untrained personnel. Because the etcd is the central backup storage for the entire Kubernetes cluster, bad planning means that there is no backup available in case of data loss due to e.g. Power outages.		
Rating: A bad planning of the etcd could occur that entire memory for configuration data of the Kubernetes cluster is not available. In case of a system failure, access to the backup memory would not be possible. The result would be a permanent loss of the data. Such a loss of all data would have catastrophic consequences for business continuity. Accordingly, this risk is to be classified as high.		

Table 11: Risk: Compromising the Nodes.

Kubernetes-System (*Confidentiality: high, Integrity: high, Availability: high*)		
Threat: **Compromising the Nodes**	Impaired Core Values: Confidentiality, Integrity, Availability	
Frequency of Occurrence without add. Measures: frequently	**Extent of Damage** without add. Measures: considerable	**Risk** without add. Measures: high
Description: An attacker who gains access to one or more worker nodes could view the data processed by the containers, manipulate them or shut down entire containers.		
Rating: A compromise of the worker nodes would affect all three core values. If an attacker compromises the worker node that includes a customer database, the attacker would have access to all customer data and he would also be able to change or even delete it. Such an attack would be existence-threatening.		

Table 12: Risk: Unauthorized Access to the etcd.

Kubernetes-System (*Confidentiality: high, Integrity: high, Availability: high*)		
Threat: **Unauthorized Access to the etcd**	Impaired Core Values: Confidentiality, Integrity, Availability	
Frequency of Occurrence without add. Measures: frequently	**Extent of Damage** without add. Measures: considerable	**Risk** without add. Measures: high
Description: Unauthorized access to the etcd can cause considerable damage to all three core values. The master node represents an attractive target for attackers, because the etcd is located in it. The etcd represents the backup storage for the entire Kubernetes cluster. Therefore, the security of the etcd is decisive for the confidentiality, integrity or availability of all objects in the Kubernetes cluster.		
Rating: Unauthorized access to the etcd would enable the attacker to view, manipulate or delete all data in the Kubernetes cluster. Such an attack would be existence-threatening.		

able for automatable attacks on the system as soon as a corresponding vulnerability becomes known.

Furthermore, the risks described are also not included in the NIST guideline ”Application Container Security Guide” from which the BSI module borrows. Thus, our findings from the IT-Grundschutz can be directly transferred to the cyber security Framework (NIST, 2017).

5 CONCLUSION

Orchestrated container virtualization environments such as Kubernetes/Docker offer a flexible, modern approach to build complex applications. Containers can be quickly assembled from preconfigured images, and the orchestration automates the management of large numbers of container instances across many hosts. However, this approach brings new challenges for the protection of database applications.

In this work, we analyzed the BSI module SYS.1.6 ”Container”, which was extended for orchestration in March 2020. For this purpose, we modeled a complex Kubernetes/Docker scenario. This scenario consisted of three containers with (i) a Postgres DBMS, (ii) the business logic and a payment system, and (iii) a web application on an Apache web server. For this scenario we derived a standard protection and a risk analysis according to BSI IT-Grundschutz.

Our analysis has shown that the module is suitable to secure a complex 3-tier business application in an orchestrated container environment. However, we were able to identify three technology-independent, additional threats that are not considered in SYS.1.6.

REFERENCES

A., C. (2019). 9 Best Kubernetes Alternatives For DevOps Engineers. https://analyticsindiamag.com/9-best-kubernetes-alternatives-for-devops-engineers/, accessed Nov. 2020.

Baier, J. (2017). *Getting Started with Kubernetes*. Packt Publishing.

BSI (2011). Taking advantage of opportunities – avoiding risks. *https://www.bsi.bund.de/EN*. Bundesamt für Sicherheit in der Informationstechnik.

BSI (2014). BSI Standards and Certification. *https://www.bsi.bund.de/EN/Topics/ITGrundschutz/itgrundschutz_node.html*. Bundesamt für Sicherheit in der Informationstechnik.

BSI (2017a). BSI-Standard 200-2: IT-Grundschutz-Methodology. *https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/International/bsi-standard-2002_en_pdf.html*. Bundesamt für Sicherheit in der Informationstechnik.

BSI (2017b). BSI-Standard 200-3: Risk Analysis based on IT-Grundschutz. *https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/International/bsi-standard-2003_en_pdf.html*. Bundesamt für Sicherheit in der Informationstechnik.

BSI (2019). BSI IT-Grundschutz-Compendium Edition 2019. *https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/International/bsi-it-gs-comp-2019.html*. Bundesamt für Sicherheit in der Informationstechnik.

BSI (2020). SYS.1.6 Container.

Docker Inc. (2020a). Docker Hub. https://hub.docker.com, accessedNov.2020.

Docker Inc. (2020b). Docker Overview. https://docs.docker.com/engine/docker-overview, accessedNov.2020.

Foundation, T. A. S. (2020). Apache Mesos. http://mesos.apache.org/.

Gao, X., Gu, Z., Kayaalp, M., Pendarakis, D., and Wang, H. (2017). Containerleaks: Emerging security threats of information leakages in container clouds. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 237–248. IEEE.

Haar, C. and Buchmann, E. (2019). It-grundschutz für die container-virtualisierung mit dem neuen bsi-baustein sys. 1.6. In David, K., Geihs, K., Lange, M., and Stumme, G., editors, *INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik – Informatik für Gesellschaft*, pages 479–492, Bonn. Gesellschaft für Informatik e.V.

Joseph McKendrick (2018). 2019 IOUG Databases in the Cloud Survey. Unisphere Research.

Kanchanadevi, P., Kumar, V. A., and Kumar, G. A. (2019). Optimal resource allocation and load balancing for a container as a service in a cloud computing. *Journal of Critical Reviews*, 7(4):2020.

Ltd., C. P. (2006). Cloudify Cutting Edge Orchestration. https://cloudify.co/.

Mardan, A. A. A. and Kono, K. (2020). When the virtual machine wins over the container: Dbms performance and isolation in virtualized environments. *Journal of Information Processing*, 28:369–377.

NIST (2017). Application Container Security Guide. *https://doi.org/10.6028/NIST.SP.800-190*. National Institute of Standards and Technology.

NIST (2020). SP 800 series on Information Security and Cybersecurity Practice Guides. https://csrc.nist.gov/publications/sp800. National Institute of Standards and Technology.

Oliveira, C., Lung, L. C., Netto, H., and Rech, L. (2016). Evaluating raft in docker on kubernetes. In *International Conference on Systems Science*, pages 123–130. Springer.

Sayfan, G. (2018). *Mastering Kubernetes: Master the art of container management by using the power of Kubernetes, 2nd Edition*. Packt Publishing.

Services, A. W. (2020). AWS Fargate. https://aws.amazon.com/fargate/.

TechnologyAdvice (2020). What is Orchestration? https://www.webopedia.com/TERM/O/orchestration.html,accessedNov.2020.

The etcd authors (2020). etcd. https://etcd.io/, accessed Nov. 2020.

The Kubernetes Authors (2020). Kubernetes Overview. https://kubernetes.io/de/docs/concepts/overview/what-is-kubernetes/,accessedNov.2020.

Vaughan-Nichols S. (2020). Kubernetes jumps in popularity. https://www.zdnet.com/article/kubernetes-jumps-in-popularity/,accessedNov.2020.