Proc. of The Fifth Intl. Conf. On Advances In Computing, Control And Networking - ACCN 2016

ISBN: 978-1-63248-104-7 doi: 10.15224/ 978-1-63248-104-7-19

Multifaceted Trust Assessment Framework for Container based Edge Computing Platform

Ehsan Mostajeran, Mohammad Fairus Khalid, Mohd Nizam Mohd Mydin, Bukhary Ikhwan Ismail, Khalid Mohammad Saleem, Hong Ong

***Abstract*—In Edge Computing platform, services are decentralized from cloud center to the edge of the platform supporting on-premises and latency sensitive applications. For this reason, traditional security mechanisms are not suitable to ensure trustworthiness of distributed and multi-tenant attributes of edge platform. Similarly by utilizing Docker as edge server, trustworthiness of host, containers and service becomes a high priority issue. Therefore in such platform run-time security assessment of the edge stack must be applied to ensure trustworthiness of resources and service. In this paper, we propose multifaceted trust framework to quantify trustworthiness of the edge servers based on security assessment of the edge stack. A chain measurement is applied to assessment process to allow insurance of the effect of each edge server's trustworthiness history on the container orchestration.**

***Keywords*—Edge Computing, Trust Framework, Docker**

I. Introduction

Cloud computing gradually developed within past few years. It has been among the top most adopted technology since its appearance [1]. The idea to centralize computing resource and storage has been solution for many of enterprises such Google, Twitter, Facebook, etc. to handle big data and its processing. However the only current cloud architecture may not be technically capable of data processing oriented scenarios with reasonable cost [2]. Processing huge amount of data produced at the edge and on-premise, bandwidth-intensive and latency-sensitive attributes of today's applications [3] are reasons to call the idea of distribution of computing resource. Therefore Edge Computing [4] [5] is proposed as a distributed cloud computing concept. The distribution of resources and computing by pushing cloud service to the edge is thought to make compatible architecture with the above matters at the user side [2] while dealing with huge amount of data wand required processing.

Ehsan Mostajeran, Mohammad Fairus Khalid, Mohd Nizam Mohd Mydin, Bukhary Ikhwan Ismail, Khalid Mohammad Saleem, Hong Ong

Advanced Computing Lab, MIMOS Berhad
Kuala lumpur, Malaysia

Logically as a matter of cost it is more optimized to push cloud service to the edge instead of sending big data to cloud from the edge, and process it, and then return the required information back to the user. So not only edge computing is a promising concept to help today's enterprise requirements, it is capable to be a fundamental layer for emerging technologies like IoT and Smart Cities. Using edge computing concept with adoption of micro services and container technology enables providers to have faster and flexible services deployment at user level, at the edge level of IoT platform [6] or any other edge-oriented architecture to improves service quality and response time [2].

Resource virtualization technology in different scales has been solution for all size of companies to cut cloud platform expenses. Currently this technology allows virtualizing several types of resources such as hardware, operating system, application, storage and network. Recently container technology (process level virtualization) has extremely become popular. Although container technology is used since year 2000, (released by FreeBSD), but recently Docker [7] introduced easier way of deployment and shipment of virtual containers which delivers huge advantages to cloud service providers in case of cost and resource allocation. Creating an abstraction of host resources such as namespaces and file system, etc. [8] allows larger scale container deployment ever [9], and public/private registries [10] to store images eases distribution of build. Using libraries such as libcontainer [11] to integrate with virtualization facility given by Linux kernel has been replaced with traditional Linux Container library [12] in Docker daemon. Such library is used to create and run containers with isolated shared kernel operation environment. This offers faster deployment of services in larger scale with no environment dependency.

Since Docker first release, it has incredibly improved cloud service provisioning [13]. Ability of caching builds improves automation of service deployment. Containers consisting services are deployed using of host resources abstractions which makes deployment faster and more efficient in term of resource allocation. Containers technology is also supported by check pointed and restored, and easy migration between clusters. Data interaction between container and volumes are more sophisticated now. But with all these benefits, trustworthiness and security of container technology has been always questioned. Docker recent plans on security and integrity of deployment, such as Notary [14], and

Proc. of The Fifth Intl. Conf. On Advances In Computing, Control And Networking - ACCN 2016

ISBN: 978-1-63248-104-7 doi: 10.15224/ 978-1-63248-104-7-19

authentication and authorization plugins [15] are not yet sufficient to say we have secure and trusted engine and containers. There are several security bottlenecks of current Docker engine and containers available. For instance kernel namespaces have been targeted widely since appearance of Docker [16]. Gaining privilege from unprivileged container in different ways by unauthorized user to compromise host and its running processes is easier when host resources like kernel is shared with containers.

To handle the above problem, trust assessment is one of the solutions that provide trusted and secure deployment workflow [17]. It is used to verify trustworthiness by verifying security state of edge devices. The state of each device where containers and services are deployed must be assessed, and evidence of host state reported back before deployment of containers and service. In Edge computing, by utilizing Docker as its infrastructure platform, trustworthiness of host, containers and service becomes a high priority issue. Hosts at the remote location must be trusted before deploying service. The question arises on how to measure trustworthiness of host, docker engine, containers and services in an environment where environment is multi-tenant and multiple separate application and services are hosted on the same edge server with shared kernel which logically increases the risk of service provisioning [5] [8].

In this work the goal is to provide a trust management framework to integrate with container provisioning engine for state evaluation of edge servers supporting a solid and secure foundation for container based workload. This system is able to identify vulnerability of edge server security and configuration, and then categorize host machines based on level of trust. The evaluation process included multiple factors of risk, reliability and credibility to perform chain state measurement to obtain host's state of trust. The rest of this paper is organized as follows.

II. Recent works

Trust in computing systems is a complex subject. From its general concept to more specific scope of trust, for instance trust in edge computing, can be built on several factors. Factors like policy, reputation, history, feedback and recommendation are examples that have been used recently to build trustworthiness of computing systems [18] [19]. With current rapid adoption of container technology [20], measuring trustworthy of services on container that deployed on shared kernel is more challenging than ever.

Trustworthy of services on top of a shared kernel is as important as trustworthy of whole cloud platform. According to new generation of threats [16], in a multitenant environment, a strong identification and authentication control, integrity of resources and real time auditing for vulnerabilities. Applying above necessary preparations may allow us to conclude that service provisioning is trusted but with the minimum threshold. Within recent years of Docker release, enterprises have achieved different approaches to build trustworthiness of containers and services running inside containers. One concept is that trustworthiness of service provisioning is achievable based on identification and secure communication by using TLS [21] between all entities. In Docker Universal Control Plane (UCP), CloudFlare's PKI and TLS toolkit CFSSL [22] is used to handle key and certificates among components in platform. In such condition, it is denotable that service is provisioned by trusted authority. The concept of root of trust is applied to keys and certificates. Moreover trust can be obtained by image author verification. Signing images is done by Notary [14] upon pushing images to registry. This indicates containers are deployed from images published by verified publishers. Trustworthiness also can be build based on trusted hardware and chain of trust like Intel and OpenStack trusted containers technology [23] ensuring service trustworthiness and integrity. But this method has limitation of availability of resources, Trusted Platform Module (TPM) [19]. The other way to implement trust is to build software layer [24] to check integrity of images and running containers. Twistlock defense system enables separate access to Docker commands by authorization plugin [15].

Basically a secure and trusted deployment workflow is needed in order to have secure and trusted containers. One main component participating in this workflow should be a standalone service with concept of Container Deep Inspection (CDI) [25] for vulnerabilities, authorization capabilities and runtime defense based on storage, memory space and networking, etc. at three layers of host, daemon and container as well as registry. Remote vulnerability assessment and state attestation has been used in order to ensure trustworthiness of host machines [26]. There are some security and configuration assessment approaches [27], but lack of a framework that run-time checks host, daemon and running containers vulnerabilities based on security configuration (where risk of each element is assigned precisely) is detected. IBM Bluemix vulnerability advisor [28], Clair by CoreOS [29] and vulnerability assessment by Flawcheck [30] are implemented but operational entity is limited to images to scan Common Vulnerabilities and Exposures (CVE) only. Dealing with security and configuration of each edge server requires run-time risk and reliability assessed. Both likehood and impact of each security parameter of host, daemon and containers should be well analyzed and operation environment should not be limited to image or container deployment level only. Meanwhile, in container based cloud platform, trustworthiness of containers and host machines cannot build based on single factor. Multifaceted trust management is more capable to be applied based on security and configuration elements.

III. Multifaceted Trust Framework

Today trustworthiness of service running on top of shared kernel containers in cloud edge computing platform is more challenging than ever. Highly distributed cloud resources at edge raises new issues of security and trust that a centralized cloud platform may not have. From service provider and consumer perspectives, trustworthiness of service provisioning is mandatory to be applied. Ensuring trustworthiness is a complex issue with wide range of categories. In our framework, trustworthiness of container based cloud service provisioning is built based on host machine, daemon and

Proc. of The Fifth Intl. Conf. On Advances In Computing, Control And Networking - ACCN 2016

ISBN: 978-1-63248-104-7 doi: 10.15224/ 978-1-63248-104-7-19

running containers security verification. Once process of assessment is applied, main functions are called to measures the state of machine according to the rule set defined by administrators. Output of each function is evidence to define the state of the device. In this work, measurement is multifaceted and consists of four main factors of risk, reliability, trust and credibility. Chain of measurement is applied between functions to keep interconnection between each state measurement. Once early stage assessment functions are called, so then trustworthiness of host machine is measured based on evidence obtained from those functions. Periodical and event-based measurements are implemented to apply effect of history of host machine state in every time trust measurement.

Adopting Center for Internet Security (CIS) Docker security scripts [31] into our framework to function as rule set for measuring current security configuration state of host machine. This measurement is applied in three layers of host, daemon and containers. Rule sets are customizable and allow adding new parameters based on requirements. Each security configuration elements of host, daemon and container is called Ej which outputs a Boolean value. When rule set is applied, result is sent to chain measurement. Different functions in chain measurements are applied which are discussed in next sections.

A. *Preliminaries*

System model (distributed remote assessment). Assessment is a protocol between Edge Prover (EP) and Central Verifier (CV) where may have separate sets of policies required for measurement process. EP is applied with system security configuration in different layer of host H, daemon D and container C. Assume we have set of security parameter E indicates {W, P} meaning warn or pass. State and integrity of device is measured based on collected evidence. So freshness of evidence is required to reflect current state of system. Evidence collected from device must include all necessary parameters. Assessment process may or may not be based on policy. So each set of evidence may be differently measured according to policy. All the assessment process must be performed based on trustworthiness evidence and protocol. Figure 1 shows system model context diagram.

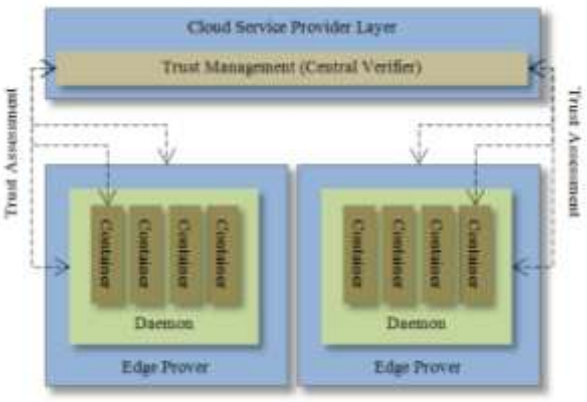

Figure 1. Graphical view of system model

Terminology. *Pair of devices (client-server) that are engaged in process of assessment are called assessment devices. Assessment is a process to query state or integrity of remote device based on collected evidence. In order to verify state or integrity of remote device, collecting and measuring sets of parameters as evidence from the remote device is performed. The process of verification of collected evidence from remote device is called measurement process. Collected evidence requires a protocol to convey evidence to verifier* [32].

Definition 1. Let ReSys(t) and RiSys(t) to be reliability and risk functions of system at time t obtained from measuring security parameters. EP is an Edge Prover that is assigned to specific set of policies. State state(EP) of each edge measures reliability and risk at three layers of host, daemon and container of edge servers. Trust value is then obtained from reliability and risk of each time state assessment. Trust function Tr(t) is obtained based chain measurement between current trust Tr(t) state and credibility Cr(t) functions at time t. Credibility presents a quantitative value of each edge device trustworthiness history.

Definition 2. Let assessment function A(t) at time t a set of assessments Ai, state measurement Si, trust Tr(t) and credibility Cr(t) function, so we have A(t)={Ai, Si, Tr(t), Cr(t)} at time t. A cycle to keep assessment updated denotes A(t=i)={Ai(t=i), Si(t=i),{ Ai(t=i-1), Si(t=i-1), Tr(t=i-1), Cr(t=i-1)}}.

B. *Assessment Factor*

1- Reliability function

Reliability of host machine affects reliability of service provisioning of edge servers. In regard to reliability [33], we consider two methods for this factor. The first method is ReCont(t) reliability of each running containers at time t. Once reliability of each container is obtained, then reliability of all containers is calculated by $\prod_{i=1}^{N} P(Ci)$ [34], P(Ci) is result from rule set applied on running containers with value total PASS.

The second method of this factor is reliability of system which is obtained from reliability of three layers of host, daemon, and container security and configuration. To calculate reliability of system, firstly we must obtain average security assessment of all containers. This is prerequisite value for calculating reliability of system. Therefore average reliability of running containers Pave(C) is obtained by $\sum_{i=1}^{N} P(Ci)/N$. In this equation, P(Ci) is result from rule set applied on running containers with value total PASS and N is number of running containers. Assigning weightage to different software stack is required in calculating reliability of system to define importance of each layer. We assumed 3 level of importance for host, daemon and container security evaluation. Reliability of system at time t ReSys(t) is then calculated by $Pave(C) * P(H) * P(D)$. In this equation P(H) and P(D) values total number of pass obtained from host and daemon layer assessment.

Proc. of The Fifth Intl. Conf. On Advances In Computing, Control And Networking - ACCN 2016

ISBN: 978-1-63248-104-7 doi: 10.15224/ 978-1-63248-104-7-19

2- Risk function

Security of Docker daemon and running container must be evaluated in a dynamic manner. Each of the security elements must be analyzed according to impact and possibility of occurrence. In this work, the likehood and impact of all rule set elements are investigated. According to element's likehood, weightages are assigned to each elements of rule set to define risk impact of element, shown in Table 1 and 2. Chain state measurement generates quantitative risk value from rule set. Either quantitative or qualitative risk evaluation requires impact of element and its likehood [35].

TABLE 1. LIKEHOOD AND IMPACT VALUE INITIALIZATION

Likehood (probability)	Impact		
	Low	Medium	High
Low	1	3	5
Medium	2	5	8
High	3	7	11

TABLE 2. LIKEHOOD AND IMPACT VALUE INITIALIZATION OF DAEMON ELEMENTS BASED ON CIS DOCKER BENCHMARK [31]

Element (Ej)	Value (σ_j)	Element (Ej)	Value (σ_j)
Do not use lxc execution driver	1	Setup a local registry mirror	8
Restrict network traffic between containers	11	Do not use the aufs storage driver	8
Set the logging level	5	Do not bind Docker to another IP/Port or a Unix socket	11
Allow Docker to make changes to iptables	7	Configure TLS authentication for Docker daemon	11
Do not use insecure registries	11	Set default ulimit as appropriate	8

Table 2 presents an example of assigning weightages to elements according to impact of each elements defined by [31]. By having these values, we propose two methods in order to calculate risk. The first method is to calculate risk of each running container by using $\sum_{j=1} \sigma jEj / \sum_{j=1} \sigma j$. Risk of each running container is obtained from result of each container security evaluation and its weightage or impact [36]. In this equation, σj is weightage of each security parameter Ej.

For calculating RiCont(t) risk of all containers we propose equation $\sum_{i=1} \lambda iW(Ci) / \sum_{i=1} \lambda i$. In this equation W(Ci) indicates result from rule set values total WARN, and ʎi is weightage for each container. If the weightage of ʎi is higher means that container is more important. This feature can be used when different types of container are running. So weightage is assigned according to type of containers.

RiSys(t) is to calculate risk of system. In the equation $\alpha Wave(C) + \beta W(H) + \gamma W(D)/(\alpha + \beta + \gamma)$, Wave(C) is average security evaluation of all containers that results WARN, and α, β and γ are weightages which defines importance of each of the layers of host, daemon and container. If they all have equal importance then we need to assign equal values to these variables, for example assign 1 to each which means they are equal and have no difference in importance.

3- Trust function

In our framework, each deployment process must have verification of trustworthiness and enough credibility of host machine upon creation of new container. So in order to calculate Tr(t) trust at time t, we propose equation below based on proposed logic by [37]. In this function Cr(t) is credibility of host at time t, ReCont(t) is reliability of container, ReSys(t) is reliability of system, RiCont(t) is risk of containers and RiSys(t) is risk of system which are explained in previous sections. There are weightages defined in this equation as variables A to F to normalize trust value and to indicate importance of each element.

$$Tr(t) = \frac{ACr(t) + BReCont(t) + CReSys(t)}{D + ERiCont(t) + FRiSys(t)} \quad (1)$$

The blow conditions must be applied for equation (1).

- Submissions of all weightages should be equal to D, so A+B+C=D in order to normalization.
- Assigning values to E and F defines importance between reliability and risk. If both are assigned with same weightage then it means both have similar level of importance. If E and F are assigned with higher values than D, then it means that risk is more important for cloud admin than reliability. If D is assigned with higher weightage than E and F, then means that reliability is more important than risk.
- This equation is developed with assumption that all values of ReSys(t), RiSys(t), RiCont(t), ReCont(t) and even Cr(t) have range of value between 0 and 1. Therefore with applying the first condition, Tr(t) must also have range of value between 0 and 1.
- If we assume Cr(t) = 0, and all security evaluation parameters result WARN (all W values are equal to 1, and all P are equal to 0) therefore we should have ReSys(t)=0, RiSys(t)=1, RiCont(t)=1, ReCont(t)=0, and for trust function we should have as below, which is a logical statement.

4- Credibility function

So far we have obtained trust value from assessment of each edge server. Implementing time manner assessment provides record of trust values for each of edge servers. These records are in fact credit of each host machine in case of trustworthiness level. Therefore we proposed a function to include trustworthiness history of host machine in each time state assessment. To calculate credibility we propose equation (2) as below.

$$Cr(t) = \begin{cases} 0, & t = 0 \\ \frac{GTr(t - \Delta t) + MReCont(t) + NReSys(t)}{U + VRiCont(t) + WRiSys(t)}, & t > 0 \end{cases} \quad (2)$$

Proc. of The Fifth Intl. Conf. On Advances In Computing, Control And Networking - ACCN 2016

ISBN: 978-1-63248-104-7 doi: 10.15224/ 978-1-63248-104-7-19

The blow conditions must be applied for equation (2).

- Submissions of all weightages of G, M and N should be equal to U, so G+M+N=U.
- Assigning values to weightages of V and W are used to define the importance of RiSys(t) and RiCont(t). It means if they are assigned with same value, both RiSys(t) and RiCont(t) have same level of importance. If V and W are assigned with higher weightage than U, then means risk is more important than reliability and vice versa.
- Assigning values to weightages of G, M and N define the importance between ReSys(t), ReCont(t) and Tr(t-Δt). If M and N are assigned with equal weightage then means that ReSys(t) and ReCont(t) have same level of importance. If G is assigned with higher weightage than M and N then Tr(t-Δt) is more important than ReSys(t) and ReCont(t) and vice versa. Δt is the difference in time of evaluating trust. For example trust evaluation cycle is every hour therefore Δt = 1 hour.

IV. Trust Function Verification

In order to verify trust function, we assume assessment result of platform with random value of reliability of container is 0.8 with risk 0.1 and credibility 0.5. We investigated trust value based on risk of system to verify whether value obtained from this equation is logical. We have taken weightages of risk of system and risk of running containers as main focus of observation. We applied Hyperbolic Tangent [38] of trust function with different values for weightages of system and running containers as shown by equation (3). Hyperbolic Tangent is used to limit the range and identifies the most accurate calculation.

$$Tr(t) = \tan\left[\frac{ACr(t) + BReCont(t) + CReSys(t)}{D + ERiCont(t) + FRiSys(t)}\right] \quad (3)$$

It is expected that when risk of system approaching 1, trust value must be reduced. Therefore as shown in figure 2, in order to apply trust function, weightages for risk of system and risk of running containers must be assigned with specific range of values to obtain well-reasoned trust values in this scenario.

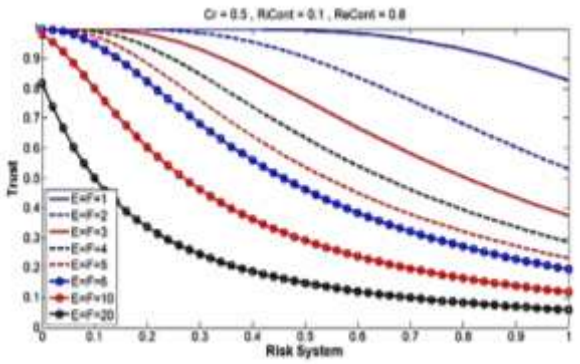

Figure 2. Trust function evaluation based on impact of security evaluation parameters

In the above equations, there are weightages that are assigned to define importance of each factor. From system administrator's perspective, importance of each factor should be defined with range of values. This is supporting the idea of allowing administrator to change importance of each factor from time to time. For example, A0 is assigned with value of X at time t0 to define high impact of credibility in trust function, but it may also require lower value of A1 = Y at time t0+1. Therefore we propose to obtain weightage of each factor at time t by A=A1+(A2-A1)tanh(A0t). Figure 2 reflects the trust value when the risk is assigned with range of weightages. Figure 3 and 4 presents decrement and increment of impact of credibility A0. We observed that A0 affects time of changing value A1 to A2. For example if the impact is assigned as A1=3, with lower value of A0, it is expected that change of impact occurs within few hours to A2=2 in case of decrement of impact of factor. And similarly is expected that if the impact is assigned as A1=2, with higher value of A0, it is expected that change of impact occurs within few hours to A2=4.

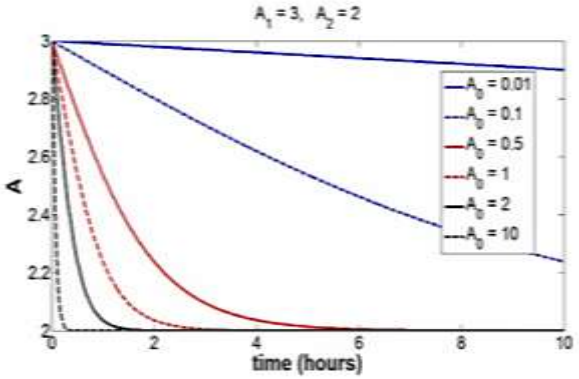

Figure 3. Evaluation of impact of factor included in trust function with decrement of impact

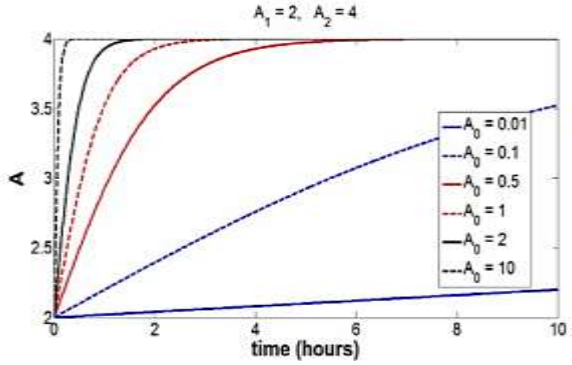

Figure 4. Evaluation of impact of factor included in trust function with increment of impact

V. Conclusion

In this paper, we propose a run-time multifaceted trust framework based on edge server's security assessment. Firstly, we have created a rule set using Center of Internet Security Docker benchmark script to apply assessment. Likehood and impact of security elements are measured, and used in risk calculation. Secondly, the result obtained from assessment process is used by framework to project trustworthy level of edge server at three levels of host, daemon and container. Lastly we used a chain measurement to ensure credibility of edge server based on trustworthy level. Using our approach in container

Proc. of The Fifth Intl. Conf. On Advances In Computing, Control And Networking - ACCN 2016

ISBN: 978-1-63248-104-7 doi: 10.15224/ 978-1-63248-104-7-19

deployment workflow, trustworthiness of Docker servers can be used as a measurement for policy management and container orchestrator to provide more trusted edge platform.

References

[1] L. Columbus, "Cloud Computing Adoption Continues Accelerating In The Enterprise," 2014. [Online]. Available: www.forbes.com/sites/louiscolumbus/2014/11/22/cloud-computing-adoption-continues-accelerating-in-the-enterprises. [Accessed: 10-Jun-2016].

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing : Vision and Challenges," *IEEE Internet Things J.*, pp. 1–10.

[3] S. Carlini, "The Drivers and Benefits of Edge Computing." Schneider Electric – Data Center Science Center, p. 8, 2016.

[4] G. Frantz, D. Freeman, and C. Link, "Extending the edge of the cloud." Texas Instruments, Dallas, Texas, pp. 1–5.

[5] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric Computing: Vision and Challenges," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, 2015.

[6] Chris Raphael, "Why IoT Edge Computing Is Crucial," *rtinsights.com*, 2015. [Online]. Available: http://www.rtinsights.com/why-edge-computing-and-analytics-is-crucial-for-the-iot/. [Accessed: 16-May-2016].

[7] "Docker: Build, Ship, and Run Any App, Anywhere." [Online]. Available: https://www.docker.com/. [Accessed: 15-Jun-2016].

[8] T. Bui, "Analysis of Docker Security," *arXiv Prepr. arXiv1501.02967*, 2015.

[9] M. F. Bin Khalid, B. I. Bin Ismail, and M. N. M. Mydin, "Performance Comparison of Image and Workload Management of Edge Computing Using Different Virtualization Technologies," in *3rd International Conference on Computer, Communication and Control Technology*, 2016.

[10] Docker, "Docker Registry." [Online]. Available: https://docs.docker.com/registry/. [Accessed: 16-Jun-2016].

[11] Docker, "libcontainer." [Online]. Available: https://github.com/opencontainers/runc/tree/master/libcontainer. [Accessed: 16-Jun-2016].

[12] I. Melia, S. Puri, K. Owens, K. Thirumalai, S. Yellumahanti, L. Herrmann, M. Coggin, J. Fernandes, K. Craven, and D. Juengst, "Linux Containers: Why They're in Your Future and What Has to Happen First." Cisco, p. 11, 2014.

[13] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers," *Performance Analysis of Systems and Software (ISPASS), IEEE International Symposium On. IEEE*. pp. 171–172, 2015.

[14] Docker, "Getting started with Docker Notary," 2016. [Online]. Available: https://docs.docker.com/notary/getting_started/. [Accessed: 05-May-2016].

[15] Twistlock, "Docker Authorization Plugin," 2015. [Online]. Available: https://github.com/twistlock/authz. [Accessed: 18-Jun-2016].

[16] Aaron Grattafiori, "Understanding and Hardening Linux Containers." NCC Group, pp. 1–122, 2016.

[17] Cam Parry, "Docker Continuous Delivery Workflows." [Online]. Available: http://capgemini.github.io/devops/docker-ci-workflows/. [Accessed: 05-May-2016].

[18] D. Artz and Y. Gil, "2007-A survey of trust in computer science and the semantic web.pdf," *Web Semant. Sci. Serv. Agents World Wide Web 5.2*, pp. 58–71, 2007.

[19] F. Corradini, F. De Angelis, F. Ippoliti, and F. Marcantoni, "A Survey of Trust Management Models for Cloud Computing," in *Proceedings of the 5th International Conference on Cloud Computing and Services Science (CLOSER)*, 2015, pp. 155–162.

[20] "8 surprising facts about real Docker adoption," *DataDog*, 2015. [Online]. Available: https://www.datadoghq.com/docker-adoption/. [Accessed: 15-May-2016].

[21] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3." Network Working Group, 2016.

[22] CloudFlare, "CFSSL: CloudFlare's PKI and TLS toolkit." [Online]. Available: https://github.com/cloudflare/cfssl. [Accessed: 10-Apr-2016].

[23] R. Yeluri and A. Gupta, "Trusted Docker Containers and Trusted VMs in OpenStack." OpenStack Summit, Vancouver, 2015.

[24] M. Bartock, P. Cichonski, J. Morello, and R. Yeluri, "DevOps and Container Security." Cybersecurity Innovation Forum, 2015.

[25] Scott McCarty (fatherlinux), "What is Deep Container Inspection (DCI) and Why is it Important?," *Redhat*. [Online]. Available: http://rhelblog.redhat.com/2015/09/03/what-is-deep-container-inspection-dci-and-why-is-it-important/. [Accessed: 05-May-2016].

[26] Open Source Security Compliance Solution, "Manages continuous scans of your infrastructure." [Online]. Available: https://github.com/OpenSCAP/openscap-daemon. [Accessed: 05-May-2016].

[27] "Docker Security Tools: Audit and Vulnerability Assessment," *Alfresco Software Ltd*, 2015. [Online]. Available: https://www.alfresco.com/blogs/devops/2015/12/03/docker-security-tools-audit-and-vulnerability-assessment/. [Accessed: 05-May-2016].

[28] Jim Doran, "Is your Docker container secure? Ask Vulnerability Advisor!," *IBM*. [Online]. Available: https://developer.ibm.com/bluemix/2015/07/02/vulnerability-advisor/. [Accessed: 07-May-2016].

[29] "Vulnerability Static Analysis for Containers," *CoreOS*, 2015. [Online]. Available: https://github.com/coreos/clair. [Accessed: 16-Jun-2016].

[30] A. Bettini, "Vulnerability Exploitation in Docker Container Environments." FlawCheck, Black Hat Europe, 2015.

[31] "Cis Security Benchmarks." Center of Internet Security, 2015.

[32] T. Abera, N. Asokan, L. Davi, T. Darmstadt, F. Koushanfar, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "INVITED Things, Trouble, Trust: On Building Trust in IoT Systems," in *Proceedings of the 53rd Annual Design Automation Conference. ACM*, 2016.

[33] R. E. Barlow and F. Proschan, *Statistical theory of reliability and life testing*. Rinehart & Winston Inc., 1975.

[34] R. Billinton and R. N. Allan, *Reliability evaluation of engineering systems- Concepts and techniques*. Springer Science & Business Media, 2013.

[35] T. Grandison and M. Sloman, "Trust Management Tools for Internet Applications," in *International Conference on Trust Management*, 2003.

[36] H. Joh and Y. K. Malaiya, "Defining and assessing quantitative security risk measures using vulnerability lifecycle and cvss metrics," in *international conference on security and management (SAM)*, 2011.

[37] James M. Sharpe and Charles H. Green, "A Note on Trust," vol. 62, no. 9, pp. 1904–1905, 2013.

[38] R. Doerfler, "Fast Approximation of the Tangent, Hyperbolic Tangent, Exponential and Logarithmic Functions." 2007.