www.czasopisma.pan.pl www.journals.pan.pl
INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2019, VOL. 65, NO.2 , PP. 211-216
Manuscript received March 18, 2019; revised April, 2019. DOI: 10.24425/ijet.2019.126303

Security Assurance in DevOps Methodologies and Related Environments

Grzegorz Siewruk, Wojciech Mazurczyk, and Andrzej Karpiński

***Abstract*—The biggest software development companies conduct daily more than hundreds deployments which influence currently operating IT (Information Technology) systems. This is possible due to the availability of automatic mechanisms which are providing their functional testing and later applications deployment. Unfortunately, nowadays, there are no tools or even a set of good practices related to the problem on how to include IT security issues into the whole production and deployment processes. This paper describes how to deal with this problem in the large mobile telecommunication operator environment.**

***Keywords*—IT security, devops, cloud security**

I. Introduction

PRESENTLY, more and more organizations are deciding to move their assets into the cloud environments. Current market conditions clearly justify this trend. It is more than enough to support this hypothesis just by analyzing the offers of cloud services by companies like Google and Amazon which provide solutions, such as Google Cloud Platform or Amazon Web Services [10]. There are also plenty of other platforms which are being offered by smaller companies working in Infrastructure as a Service (IaaS) [10] models which are based, for example, on the Open Source OpenStack [9] platform.

This trend is caused by many factors, such as the price of such service or possibility to integrate most popular project methodology i.e. an agile development of IT system life cycle [4]. Currently, more and more teams are deciding to work in DevOps [1] methodology (clipped compound of the Development and Operations) which combines software development with information technology operations. The main goals of these two approaches are to shorten system development life cycle while delivering features, fixes and updates. Thanks to the mentioned cloud infrastructure architecture all together (methodology and tools) are being easily integrated with each another.

In this paper we will introduce a novel approach to managing security of IT systems, which is based on the metric that allows to evaluate in real-time manner the security level of an IT system (or particular change during patching process) and evaluates if it meets requirements which are described in Section V.

The rest of the paper is structured as follows. Section II briefly introduces the most relevant existing works related to the topic of this paper. Then in Section III DevOps and cloud computing concepts are described with the special focus on their security aspects. Next, in Section IV solutions that enable measuring security levels are characterized, while in Section V a novel metric to express this level is described. Finally, in Section VI a case study is presented to prove the usefulness of the proposed approach, and Section VII concludes our work.

Grzegorz Siewruk is with Warsaw University of Technology, Institute of Telecommunications, Faculty of Electronics and Information Technology and Orange Poland, Department of ITN Security, Warsaw, Poland (e-mail: *g.siewruk@tele.pw.edu.pl*)
Wojciech Mazurczyk is with Warsaw University of Technology, Institute of Telecommunications, Faculty of Electronics and Information Technology, Warsaw, Poland (e-mail: *w.mazurczyk@tele.pw.edu.pl*)
Andrzej Karpiński is with Orange Poland, Department of ITN Security, Warsaw, Poland (e-mail: *andrzej.karpinski@orange.com*)

II. Related Works

There are many works which are focused on investigating the level of security for the cloud infrastructure environment, e.g., [3], [8], [11], or [12]. Some of them are focused on creating a set of tests (benchmarks) for the infrastructure provider layer (OpenStack [10]), while other introduce interesting implementations of SAST (Static Application Security Testing) and DAST (Dynamic Application Security Testing) security scanners [2], [14]. A system which aim is to detect violations of IT Security principles, described in [17] can be used as Intrusion Detection System (IDS) in projects operating on a public cloud. The closest to the approach described in this paper is the solution presented in [19], which proposes to evaluate IT security of the cloud environment based on the defined metric which describes an acceptable level of security (which is calculated based on the number of discovered vulnerabilities). The complete process of security management for the mentioned work is illustrated in Fig. 1.

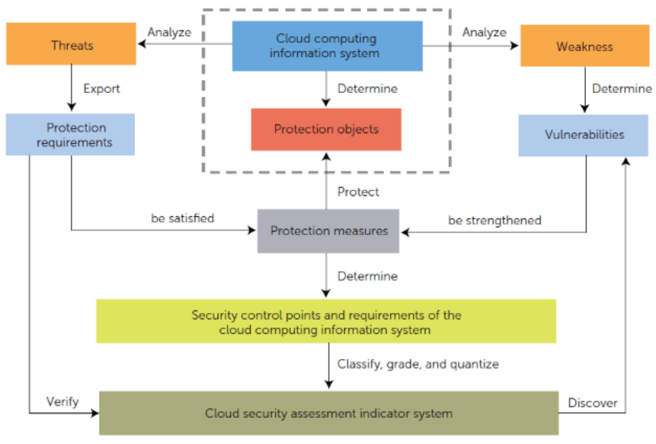

Threats
Analyze
Cloud computing information system
Analyze
Weakness
Export
Determine
Determine
Protection requirements
Protection objects
Vulnerabilities
Protect
be satisfied
Protection measures
be strengthened
Determine
Security control points and requirements of the cloud computing information system
Classify, grade, and quantize
Verify
Cloud security assessment indicator system
Discover

Fig. 1. Security Management in [16]

The proposed system architecture is a good starting point from where IT Security evaluation of the cloud environment could be made. It covers multiple areas which should be taken into consideration during threat management process. It gathers information about the system components as well as requirements for the configuration and vulnerabilities. Unfortunately, in order to create fully automated process there also has to also available an information about the context which should describe application running inside particular system ecosystem.

Each of the above mentioned solution lacks application context and environment in which the application is deployed. In this work we would like to fill this gap by proposing a metric which will take both – the application-level context and the application run time environment into consideration.

III. DEVOPS AND PUBLIC CLOUD SECURITY

DevOps methodology was presented for the first time in 2009 [1] but its real expansion can be observed during the last few years. This trend is related to the fast development of tools within the class Continuous Integration / Continuous Deployment (CI/CD) which are ideally designed to ensure a proper level of tasks automation, such as virtual machine deployment or running newly created source code onto proper machine. Discussing the details of the DevOps model is outside the scope of this paper, but in a nutshell its essence is related to treating the whole infrastructure as a code written in ansible or terraform [12]. Both tools are opensource software that automates software provisioning and configuration management. This approach allows to prepare the piece of a source code that is responsible for configuring the virtual machine, configuring firewall rules, pulling the source code from the repository and then start the web application server. Taking IT security into consideration, such an approach has many advantages like recurrence of operations (a user may forget to implement one of hundreds firewall rules, however once prepared program cannot), homogeneity of the environment and the speed of action. Considering above, there are also disadvantages to be acknowledged, such as – broad range of permissions given to a tool that is widely available. Critical vulnerability found in one element (CI/CD) which contains provider configuration may put the whole platform at risk as it can be compromised**.** In the whole process of software development, many other points are related with the identified threats such as identity management, verification of images used (which are installed on the servers), software testing or checking for available updates. In each of these areas, security management is necessary and can be performed in several ways.

Development methodologies are not the only "location" where IT security should be considered. The other is the infrastructure layer. The primary goal of the OpenStack (which now is leading technology for the cloud computing infrastructure) was to create global standards for the cloud environments. Continuous need for more computing power, increased disk storage and faster than before data access have led to the development of this platform. It must be also emphasized that OpenStack has significantly evolved from the first published version of this software in 2010 which had only two modules – Nova and Swift. The version of Queens published in February 28, 2018 has as many as 39 modules. It must be also noted that a lot of new factors have been considered from the security point of view. That could compromise both software infrastructure and stored information.

Table I illustrates how the OpenStack platform has developed. More and more important features are being added by the platform itself (and can be automated), for example, managing Domain Name Server (DNS) or storing encryption keys. Each release of a new version contains new possible attack vectors, so securing such an environment should be a continuous process.

Therefore, in this case, it is crucial to perform a risk analysis to identify areas that require special attention. It should be performed to adopt appropriate security mechanisms that would minimize the probability of launching a successful attack on a system.

Note, that the security within public cloud environments (the whole or a part of so-called hybrid cloud [6]) can be described on several levels:

– *IaaS Layer* – continuous verification of settings, including a list of administrator accounts and an analysis of the permissions,

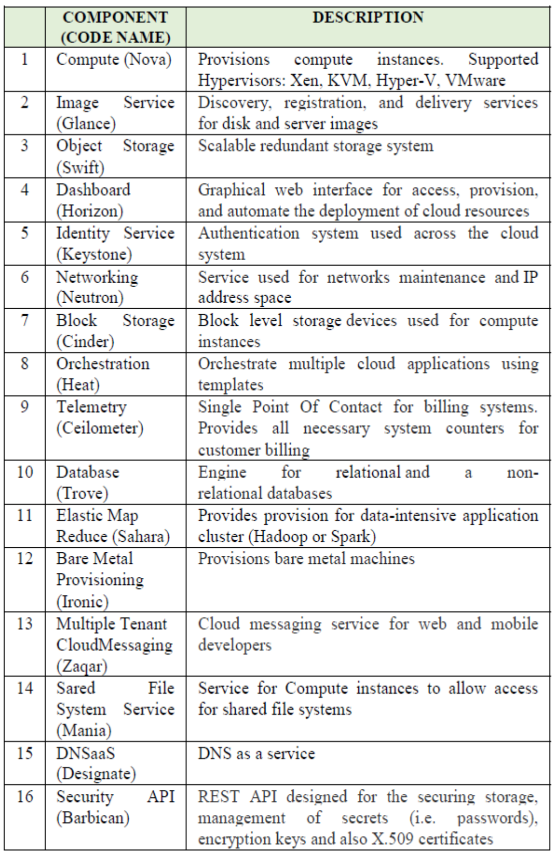
– *Operating System Layer* of running virtual machines – continuous verification of the current software updates,

– *Network Layer* – continuous verification of the method of how the system is being exposed in various network segments, verification of launched services (if there are no running ports that are typically used in the well-known network attacks), verification of whether multiple machines do not have excessive connection permissions to each other,

– *Application Layer* – continuous security tests of the web applications, mobile applications and APIs as well as source code analysis from the security perspective,

– *CI / CD Layer* – verification of the running scripts.

TABLE I
SELECTED OPENSTACK MODULES



	COMPONENT (CODE NAME)	DESCRIPTION
1	Compute (Nova)	Provisions compute instances. Supported Hypervisors: Xen, KVM, Hyper-V, VMware
2	Image Service (Glance)	Discovery, registration, and delivery services for disk and server images
3	Object Storage (Swift)	Scalable redundant storage system
4	Dashboard (Horizon)	Graphical web interface for access, provision, and automate the deployment of cloud resources
5	Identity Service (Keystone)	Authentication system used across the cloud system
6	Networking (Neutron)	Service used for networks maintenance and IP address space
7	Block Storage (Cinder)	Block level storage devices used for compute instances
8	Orchestration (Heat)	Orchestrate multiple cloud applications using templates
9	Telemetry (Ceilometer)	Single Point Of Contact for billing systems. Provides all necessary system counters for customer billing
10	Database (Trove)	Engine for relational and a non-relational databases
11	Elastic Map Reduce (Sahara)	Provides provision for data-intensive application cluster (Hadoop or Spark)
12	Bare Metal Provisioning (Ironic)	Provisions bare metal machines
13	Multiple Tenant CloudMessaging (Zaqar)	Cloud messaging service for web and mobile developers
14	Sared File System Service (Mania)	Service for Compute instances to allow access for shared file systems
15	DNSaaS (Designate)	DNS as a service
16	Security API (Barbican)	REST API designed for the securing storage, management of secrets (i.e. passwords), encryption keys and also X.509 certificates

IV. Methods for Measuring Security Level

A. IaaS Layer

Every of currently available IaaS platform has an API that allows both reading and modifying data in order to obtain full information on how resources are used. It is necessary to be able to automatically retrieve information about currently running servers along with information about addresses of interfaces and the configuration of firewalls rules. Another important aspect is the list of users who have access to the IaaS layer and their permissions. The whole functionality consists of the necessary component which is called Service Discovery module that is responsible for gathering information about assets that build system.

B. CI/CD Layer

As CI/CD tools can be described as a software designed to support whole deployment process it is hard to define it as a separate layer, but for the purpose of this paper we will do so (full automation processes are going through CI which makes it great place to put security evaluation mechanisms).

In a fully automated environment, all resources will be run by previously prepared scripts, e.g., using ansible or terraform. The mere appearance of a new virtual machine without information about running a script that creates such a machine may indicate a security breach (but it does not necessarily have to). If configured and used appropriately, the CI layer is an invaluable source of information that feeds the Service Discovery module. We are able, for instance, to obtain information on the parameters of the machine that is running or the branch of code which is currently compiled in order to be published on the server soon. And in addition to information obtained from the IaaS layer, it gives us full knowledge about what applications are running on what resources.

C. The Operating System Layer

In this case we are considering if components of the solution like operating systems are configured properly. Organizations like the Center for Internet Security (CIS [3]) are preparing a list of good practices which should be fulfilled in order to create as much secure as it is possible component. CIS is sharing security benchmarks for OSes like: Centos, Redhat, or Ubuntu. In addition, test suites are available for solutions such as Kubernetes and Docker. Proper source codes are possible to be downloaded from the Github which implements CIS requirements and checks the system on which it runs against them (requirements). The outcome is in the form of "requirement – result". The second important element to be examined is the presence of the automatic updates (at least when it comes to the updates that are marked as security fixes).

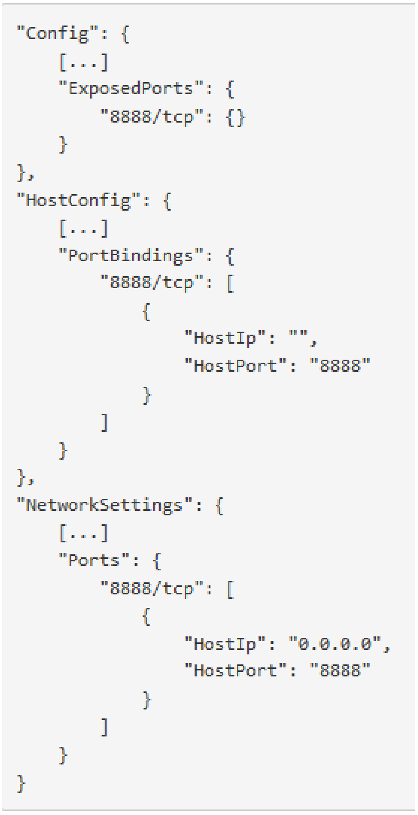
D. Network Layer

Assuming that we are in the possession of complete information on how an IT system has been built (from the Service Discovery element which gathered information about IaaS platform and obtained information from the CI/CD) it is possible to prepare a set of security tests by automated tools like Tenable Nessus or OpenSource solution like w3af [5]. The results of such a network test is complete information on how certain server is being used, i.e., information on currently opened ports and which services are listening on them. The last step to fully evaluate this layer is to compare the discovered opened ports with the intended configuration. It is possible by utilizing a configuration file which is used by the CI and is called the docker file. An exemplary docker file is presented in Fig. 3.

In this case, the application is using only port TCP "8888". If security test discovers opened TCP "80" port with an active web application server listening on it, this could be an indicator that the hardening process did not go well and it should be repeated.

E. Application Layer

If we are considering the security of applications, there are two sets of tests to be mentioned: automatic security tests (done by a specific tools) and manual penetration testing (done by a qualified expert). The latter is always more accurate (a human can try to examine context which machines typically do not understand) but in environments like the one we are mentioning in this paper (where changes are deployed frequently) it is almost impossible to implement. The main reason for this is the time which an expert needs to perform the penetration test. During the time required for performing a single test, a team of developers could prepare several changes. Thus, waiting for the test to be completed delays the release of application version which is ready to be put on production infrastructure(one of the key reasons why to switch to DevOps methodology is related to shorter releases time).



```
"Config": {
    [...]
    "ExposedPorts": {
        "8888/tcp": {}
    }
},
"HostConfig": {
    [...]
    "PortBindings": {
        "8888/tcp": [
            {
                "HostIp": "",
                "HostPort": "8888"
            }
        ]
    }
},
"NetworkSettings": {
    [...]
    "Ports": {
        "8888/tcp": [
            {
                "HostIp": "0.0.0.0",
                "HostPort": "8888"
            }
        ]
    }
}
```

Fig. 2. An exemplary configuration of the dockerfile

Automatic security tests can be done by several types of scanners: static application security testing (SAST – mostly source code analysis) which conducts a set of tests on the static source code. Unfortunately, results of the SAST scanning often contain multiple false positives, for example, identified vulnerability could be impossible to exploit in the context of running application. In contrast, the Dynamic Application

Security Testing (DAST) which conducts automatic penetration security testing, e.g., by crawling a website and use well-known web application vulnerabilities in order to evaluate its security are more accurate. But they also have some issues – from which the biggest one is the limited scope of testing scenarios which depends on type of tool used. There is also another class of automated testing tools – Interactive Application Security Testing (IAST) which combines static and dynamic testing. In this paper we do not consider IAST scanners which are available to use (both opensource and commercial ones) as they are strongly attached to the software development strategy. Most of the IAST scanners are being executed during functional unit testing (where it is known which part of the source code is responsible for the certain website functionality). However, it is crucial that the unit tests of application should be created properly for the IAST to function. The solution presented in this paper will be evaluated within the Telecommunication Provider environment where dozens of development teams work simultaneously and each one of them is using different testing strategy.

In the course of our future work, as a long-term goal, we would like to introduce an alternative to the IAST scanners which are available on the market today. Its main advantages are that it will be independent of the run time environment and it will be based on the findings from the DAST and SAST scanners. Using obtained results, the machine learning algorithms will group vulnerabilities in a way that each vulnerability found from the dynamic scan will be linked with a specific vulnerability found during the static analysis. In this way a developer will be able to get information which particular line of the source code is responsible for vulnerability in the web application, which in the end will expedite the fixing process. In order to achieve this purpose there two types of algorithms to be used. First one is to create relation between findings which came from different sources – therefore a clustering algorithm is necessary to be applied. Then in order to determine if the identified vulnerability is exploitable or it is just not an issue a classifying algorithm would be needed. In this moment of maturity of the project it is not yet decided if it will be basic implementation of k-means[13], LDA [8] or SVM[7] or modified version of those.

V. The Proposed Grading Metric

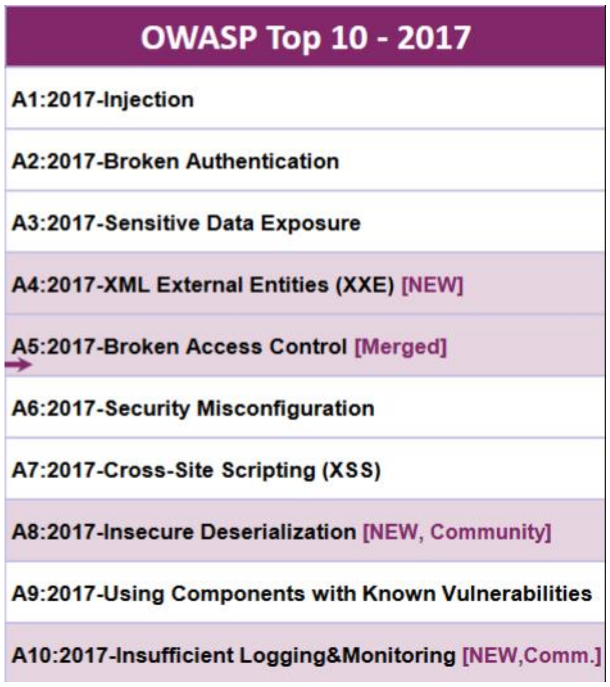
From the layers described in Section IV, first two (IaaS and CI/CD layers) can be described as part of a system responsible for providing information about the environment and assets in an automatic manner. The rest of them are related to the anomaly detection.

The quality of the created classifier can be measured with the four standard binary classification metrics:

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ which is the ratio of correctly classified events to the whole set.
- $Precision = \frac{TP}{TP+FP}$ which describes classification ability not to detect false events.
- $Recall = \frac{TP}{TP+FN}$ which describes classification ability to detect actual anomalies.
- $F1 = \frac{2*precision*recall}{precision+recall}$ which is a weighted average of the classifier precision and recall.

where, *TN* – is true negative (vulnerability which is not possible to be exploited in a described context), *TP* – true positive (vulnerability which is possible to be exploited), *FN* – false negative (vulnerability which can be exploited but is inversely marked by a classifier), *FP* – false positive (vulnerability which cannot be exploited but is marked as exploitable). Properly created classificator to be considered useful should has both precision and recall at the highest possible level..

While having one source of data and one classifier problem, we are limited just to calculating its precision and recall. When having multiple data sources (e.g. more vulnerability scanners) metrics are harder to be evaluated because each vulnerability found will have different significance (it depends on the data source, scanner and application context). It is easy to imagine that the Cross Site Scripting (XSS) vulnerability will be treated as more important threat while occurring in the web application graphic user interface (GUI) and differently while occurring in web application used only by an application programming interface (API). A good starting point could be analyzing OWASP [13] top 10 vulnerabilities report in order to describe metrics for the most popular vulnerabilities in web applications (see Fig. 3).



OWASP Top 10 - 2017
A1:2017-Injection
A2:2017-Broken Authentication
A3:2017-Sensitive Data Exposure
A4:2017-XML External Entities (XXE) [NEW]
A5:2017-Broken Access Control [Merged]
A6:2017-Security Misconfiguration
A7:2017-Cross-Site Scripting (XSS)
A8:2017-Insecure Deserialization [NEW, Community]
A9:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Fig. 3. OWASP top 10 vulnerabilities for web applications [13]

The proposed approach to calculate security level metric of the solutions deployed in the cloud environment is as follows:

$$LC * \sum_{Classifiers} \left(\frac{TP*W}{TP}\right) * P \qquad (1)$$

where,
LC – Coverage level of the security events collection,
TP – True positive count for the particular classifier,
W – Weight of the detected vulnerability,
P – Precision value for the evaluated classifier.
We believe that a single metric which indicates the level of security based on the coverage level and depends on the

classifier recall (classifier ability to detect existing security vulnerabilities) and each identified true positive value weight would be most effective in the scale of the Telecommunication Operator environments.

It is hard to find information about how other researchers are calculating this metric and even harder to obtain detailed information on how it is obtained in the commercial solutions (e.g. ThreatFix) so in the future works it will be proven by comparison that the described metric is superior when compared to other solutions.

VI. Case Study

In order to prove the usefulness of the introduced security metric, a manual analysis of the vulnerabilities for a single application has been performed. Experimental data has been obtained from the real-life security testing processes from one of the major mobile network operator in Poland. As a test object a web application built from Java microservices has been selected. A list of false positives, identified as vulnerabilities but marked by an expert as not major, and a list of true positives, treated as vulnerabilities and marked as exploitable, from the last 5 deployments in DevOps model of a given application have been prepared. Only two security issue sources were available for the presented IT system – source code security audit and CIS kubernetes and docker benchmark compliance. SAST analysis is being conducted by using MicroFocus Fortify software, and CIS benchmark compliance verified using set of scripts prepared by Github's user - dev-sec (kubernets 2.11 and docker 1.13 version). Single run described the particular deployment made on the production ecosystem. In this case study we have analyzed last 5 deployments, where run 1 is the farthest in time and run 5 is the latest one. Overall scheme of the study is presented in Fig 4.

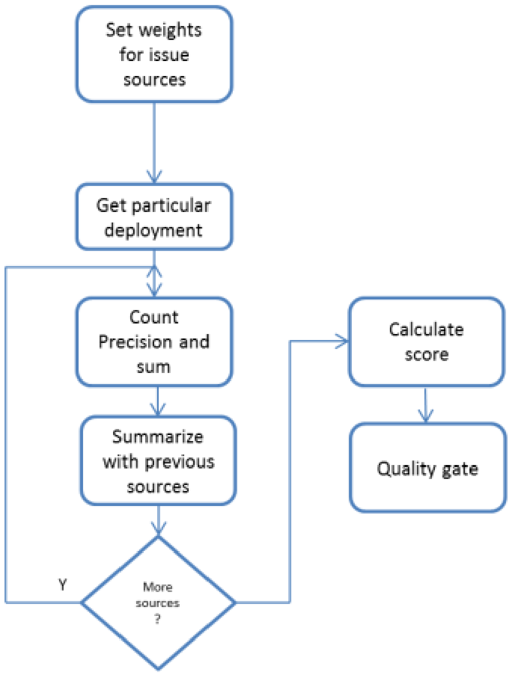

Set weights for issue sources
Get particular deployment
Count Precision and sum
Summarize with previous sources
More sources ?
Y
Calculate score
Quality gate

Fig. 4. Case study scheme

TABLE II
EVALUATION OF THE PRESENTED METRIC

Application 1							
Run	**Scanner**	**TP**	**FP**	**Weight**	**Precision**	**sum**	**score**
1	**Code**	172	807	0.5	0.18	0.09	0.33
	Audit	84	89	0.5	0.49	0.24	
2	**Code**	177	912	0.5	0.16	0.08	0.32
	Audit	82	89	0.5	0.48	0.24	
3	**Code**	**230**	**798**	**0.5**	**0.22**	**0.11**	**0.43**
	Audit	**115**	**68**	**0.5**	**0.63**	**0.31**	
4	**Code**	175	805	0.5	0.18	0.09	0.33
	Audit	81	90	0.5	0.47	0.24	
5	**Code**	164	804	0.5	0.17	0.08	0.28
	Audit	65	104	0.5	0.38	0.19	

Based on the results depicted in Table II it is possible to observe that for one particular deployment the overall score is much larger than for the other runs (by approximately 10%) – calculated using equation from Section V. Utilization of the automated security quality gate run 3 should not be allowed to be deployed in the production environment, however, without an automated process it would be impossible to be achieved. Even if the introduced vulnerabilities were fixed before the next run, there was still a brief time when newly created vulnerabilities were exploitable in the production environment. The weight for each security issue source was set equally to 0.5 (as it was decided that each source was equally important). When having more than 2 vulnerability sources a risk analysis should be performed and based on the results the weights to set should be decided. For example when the process is built from 4 security issue sources – DAST, SAST, Infrastructure scanner and compliance data it could be set as follow:

- SAST – 0.3 – in most cases the static source code analysis covers full code base so it is very accurate source of information.
- Infrastructure – 0.3 – the infrastructure scans detect vulnerabilities in the installed software and in most of the cases provide information with high severity.
- DAST – 0.25 – the dynamic tests in the described scenario are being conducted from the behind a proxy which strips requests and thus in turn makes the tests results not fully accurate.
- Compliance – 0.15 – as issues from this source are often considered as supplementary information.

On the other hand in run 2 we can see that a single increase of the FP value was not making the difference on the overall security score.

It was proven in this section that using presented security metric (section V) and implementing quality gate inside DevOps pipeline we will add possibility to block changes of application which contain security vulnerabilities in the automatic manner.

VII. Conclusion

To authors' best knowledge there are no existing methods or tools able to measure and evaluate in a complex manner the IT security levels of solutions deployed in the cloud environments. The scope and quantity of the processed data and the pace at which new environments are being built, strengthen our belief that the current development methodologies lack a generic way to calculate system overall security level. This paper proposed areas which should be taken into consideration while preparing the system architecture. Having all areas covered for the Service Discovery module, it is possible to grade the level of security of the whole solution, which depends only on the environment configuration, services running on the assets, and vulnerabilities found in the deployed software. Note, that the proposed approach does not need developers to create additional test cases.

Next step will be to implement described solution in the environment of a large telecommunication provider. The first step has already been accomplished i.e. integration with the IaaS layer. Integration with CI/CD layer is difficult enough that multiple CI tools and scripting techniques exist. When the full integration with CI/CD is accomplished, it will be possible to get complete information for the Service Discovery module and then based on this information create security scanners in order to start classifier learning process.

References

[1] Abubaker Wahaballa, Osman Wahballa, Majdi Abdellatief, Hu Xiong and Zhiguang Qin, "Toward unified DevOps model" in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS).*

[2] Adnan Masood and Jim Java, "Static analysis for web service security - Tools & techniques for a secure development life cycle" in *2015 IEEE International Symposium on Technologies for Homeland Security (HST).*

[3] *Center for Internet Security*. Downloaded from https://www.cisecurity.org/cis-benchmarks/

[4] Chirag Doshi and Dhaval Doshi, "A Peek into an Agile Infected Culture" in *2009 Agile Conference.*

[5] Daniel Ståhl, Kristofer Hallén and Jan Bosch, "Continuous Integration and Delivery Traceability in Industry: Needs and Practices" in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA).*

[6] Adam Gordon, "The Hybrid Cloud Security Professional" in *IEEE Cloud Computing (Volume: 3, Issue: 1, Jan.-Feb. 2016)* .

[7] H. Drucker, Donghui Wu and V.N. Vapnik, "Support vector machines for spam categorization" in IEEE Transactions on Neural Networks (Volume: 10 , Issue: 5 , Sep 1999)

[8] Hongchen Gui, Qiliang Liang and Zhiqiang Li, "An improved AD-LDA topic model based on weighted Gibbs sampling, 2016 IEEE Advanced Information Management" in Communicates, Electronic and Automation Control Conference (IMCEC)

[9] Ionel Gordin, Adrian Graur, Alin Potorac and Doru Balan, "Security Assessment of OpenStack cloud using outside and inside software tools" in *14th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS, Suceava, Romania, May 24-26, 2018.*

[10] Lindita Nebiu Hyseni and Afërdita Ibrahimi, "Comparison of the cloud computing platforms provided by Amazon and Google" in *2017 Computing Conference.*

[11] Marco Anisetti, Claudio A. Ardagna, Ernesto Damiani and Filippo Gaudenzi, "A Security Benchmark for OpenStack" in *2017 IEEE 10th International Conference on Cloud Computing.*

[12] Nishant Kumar Singh, Sanjeev Thakur, Himanshu Chaurasiya and Himanshu Nagdev, "Automated provisioning of application in IAAS cloud using Ansible configuration management" in *2015 1st International Conference on Next Generation Computing Technologies (NGCT).*

[13] *OWASP*. Downloaded from https://www.owasp.org/index.php/
Main_Page

[14] P. P. W. Pathirathna, V. A. I. Ayesha, W. A. T. Imihira, W. M. J. C. Wasala, Nuwan Kodagoda and E. A. T. D. Edirisinghe, "Security testing as a service with docker containerization" in *2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA).*

[15] Shruti Kapil, Meenu Chawla and Mohd Dilshad Ansari, "On K-means data clustering algorithm with genetic algorithm" in 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)

[16] Shubham Awasthi, Anay Pathak and Lovekesh Kapoor, "Openstack-paradigm shift to open source cloud computing & its integration" in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I).*

[17] Turki Alharkan and Patrick Martin, "IDSaaS: Intrusion Detection System as a Service in Public Clouds" in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.*

[18] Wu Qianqian and Liu Xiangjun, "Research and design on Web application vulnerability scanning service" in *2014 IEEE 5th International Conference on Software Engineering and Service Science.*

[19] Xuexiu Chen, Chi Chen, Yuan Tao and Jiankun Hu, "A Cloud Security Assessment System Based on Classifying and Grading" in *IEEE Cloud Computing Published by The IEEE Computer Society*.

[20] Yasuharu Katsuno, Ashish Kundu, Koushik K. Das, Hitomi Takahashi, Robert Schloss, Prasenjit Dey and Mukesh Mohania, "Security, Compliance, and Agile Deployment of Personal Identifiable Information Solutions on a Public Cloud" in *2016 IEEE 9th International Conference on Cloud Computing.*

[21] Ziqiang Zhou, Changhua Sun, Jiazhong Lu and Fengmao Lv, "Research and Implementation of Mobile Application Security Detection Combining Static and Dynamic" in *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA).*