Cloud Security Automation Framework

Cihan Tunc[1,2], Salim Hariri[1], Mheni Merzouki[2], Charif Mahmoudi[2], Frederic J. de Vaulx[2], Jaafar Chbili[2], Robert Bohn[2], Abdella Battou[2]
[1] The University of Arizona, Tucson, Arizona, USA
[2] National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA
[1]{cihantunc, hariri}@email.arizona.edu
[2]{cihan.tunc, mheni.merzouki, charif.mahmoudi, fdevaulx, jaafar.chbili, robert.bohn, abdella.battou}@nist.gov

***Abstract*—Cloud services have gained tremendous attention as a utility paradigm and have been deployed extensively across a wide range of fields. However, Cloud security is not catching up to the fast adoption of its services and remains one of the biggest challenges for Cloud Service Providers (CSPs) and Cloud Service Consumers from the industry, government and academia. These institutions are increasingly faced with threats affecting the confidentiality, integrity and availability of the cloud resources such as DoS/DDoS attacks, ransomware attacks, and data breaches to name a few. In the current cloud systems, security requires manual translation of security requirements into controls. Such an approach can be for the most part labor intensive, tedious and error-prone leading to inevitable misconfigurations rendering the system at hand vulnerable to misuse be it malicious or unintentional. Therefore, it is of utmost importance to automate the configuration of the cloud systems per the client's security requirements steering clear from the caveats of the manual approach. Furthermore, cloud systems need to be continuously monitored for any misconfiguration, and therefore lack of the required security controls. In this paper, we present a methodology allowing for cloud security automation and demonstrate how a cloud environment can be automatically configured to implement the required NIST SP 800-53 security controls. Also, we show how the implementation of these controls in the cloud systems can be continuously monitored and validated.**

Keywords—cloud computing, cybersecurity, automation, security controls

I. INTRODUCTION

Cloud services have been one of the most important paradigms of today's IT world due to its salient features of on-demand, flexible, scalable, ubiquitous computing with minimal resource management effort for the end-users. The National Institute of Standards and Technology (NIST) defines the cloud computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. The sheer fact that cloud services offer an on-demand model and therefore promote efficient spending on IT departments is reason enough for the ongoing movement to deploy cloud systems in both government and non-government institutions. Thus, per International Data Corporation (IDC), by 2018, at least half of the IT expenses will be cloud-based with a reach of 60% of all IT infrastructures and 60-70% of all software services by 2020 [2]. And, Wikibon predicts that by 2022, Amazon AWS platform by itself will be approximately $43 billion revenue per year, providing 8.2% of all cloud business [3].

Even though cloud computing is considered a major IT movement, the cloud security remains a plaguing challenge, according to a RightScale report (25% of respondents cited cloud security, lack of resources/expertise, and managing cloud spending as the main challenges) [4]. The cumulative cost of cyberattacks to an organization averages $3.5 million annually [5], which gives an economic illustration of the importance of cloud security. Stakeholders from all fields steadily realize that Cloud services face numerous threats: data breaches, compromised credentials, account hijacking, permanent data loss and DoS/DDoS attacks just to name a few [6]. Add to that the lagging CSPs security default capabilities which do not meet the organization's security and privacy requirements [7]. From the customer's perspective, more institutions from the private sector are developing interest in the NIST Cyber Security Framework (CSF) and Risk Management Framework (RMF) to address and manage security risk, define requirements and security controls implementing them. However, to the best of our knowledge, one cannot find a well-defined practical approach translating the said security controls into actionable items running on the cloud environments. Therefore, in this paper, we present a methodology to automatically create a cloud computing environment implementing NIST SP800-53 security controls to satisfy cybersecurity requirements of the cloud systems at hand.

In summary, this paper aims at answering the following questions: (a) What are the security requirements from both the user side and the CSP side? (b) How can a user specify the security requirements? (c) How can we automate the deployment of cloud systems that meet user security requirements? and (d) How can we validate that the cloud systems meet the security needs and provide a comprehensive compliance report to support?

The rest of the paper is organized as follows. In Section II, we present the background information on the cloud security and standards domains. We present a cloud computing security taxonomy in Section III to be used for better understanding and categorizing each aspect of cloud security. The proposed methodology is presented in Section IV. Next, we present a proof-of-concept implementation in Section V. Finally, Section VI concludes the paper.

II. Background

A. Cloud Computing and Security Appraoches

Cloud computing involves the delivery of services through different models such as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). While this paper mainly focuses on the IaaS model, it can be extended to other service models as well. For the IaaS, the users are offered virtual machines (VMs) residing through a hyper vision technology such as Xen or VMware among others. Due to the complex structure of the cloud services, cloud computing suffers from multiple security issues [8]. All cyber-attacks targeting physical machines in the physical domain exist on virtual machines running on the cloud environment, with added burden from the hypervisor. As the cloud offers a pool of resources that users share, the importance of the hypervisor security increases. The dependency of cloud computing on the virtualized environment raises more security issues, like hypervisor exploitations [9]. One instance of these attacks is the injection of malware in the publicly available virtual machine image, which subsequently affects the services of the cloud. Another major security issue for cloud computing systems is the insider attacks [10].

Many solutions have been proposed to solve security issues [11-14] in cloud computing. Some cloud security systems implemented a recovery-based intrusion tolerant algorithm that enhances the availability and resilience of cloud services or focused on hiding the data as a method to increase services' resilience to attacks [13]. Many of the security solutions developed for the cloud focus on efficiently protecting the cloud storage against diverse range of attacks including roll-back attacks [15]. Some of the proposed security solutions use innovative risk-based analysis for the security testing of the cloud environment. This risk-based analysis reduces the number of possible misuse cases of the cloud [14].

B. Autonomic Computing

By definition, cloud computing should have dynamic management and self-organization [1, 16, 17]. Nevertheless, most of the current cloud implementations are missing those two essential attributes. Understanding the functional requirements of cloud services is an essential key in defining the architecture of the system providing those services [18]. Applying software design approaches to the development of autonomic management systems provides a solid ground to maintain the service level agreement, protect against external attacks, prevent failures and enables recovery [19]. Proactively managing failures and providing self-healing is an important attribute of the cloud [20]. Self-configuration allows the cloud to adapt to changes by tuning the allocation of virtual resources and the applications parameters, thereby achieving self-optimization [21].

C. Cloud Standards

As cloud computing is being used ubiquitously, there are multiple cloud standards works in the literature such as the one from the Cloud Standards Customer Council (CSCC) aiming at cloud systems' successful adoption and solving the security and interoperability issues [22]. Hence, in terms of security, CSCC presented a reference for the organizations adopting the cloud computing and its security impacts [23]. Organization for the Advancement of Structured Information Standards (OASIS) provides open standards for the IT world in collaboration with the industry and academia [24]. Topology and Orchestration Specification for Cloud Applications (TOSCA) is one of OASIS standards intended to define services and applications, and their relationships, especially for cloud systems with a focus on automated management, portability and deployment [24]. For this purpose, TOSCA introduces a grammar for describing service templates, requirements, capabilities, and policies with the help of YAML. While TOSCA is a standard model, there are multiple implementations such as OpenTOSCA [25], Cloudify [26] in and OpenStack Heat [27]. However, current implementations of this standard do not address the cloud security challenges and only focus on the management and automation of the application deployment.

D. Discussion

In summary, the current efforts on cloud automation focus heavily on performance and predominantly overlook the vital security aspect. Also, to the best of our knowledge there have not been any studies with focus on automation of security mechanisms implementation in cloud systems. Therefore, this paper is the first attempt in this direction.

III. Cloud Computing Security Taxonomy

To fully understand the cloud computing security issues, we first developed a cloud security taxonomy based on NIST SP 800-53 [28] and Federal Risk and Authorization Management Program (FedRAMP) [29] security assessment framework. Next, we utilized the taxonomy to implement the required security controls and their management processes. Our security taxonomy has three sub-categories: Security components, privacy, and compliance, as shown in Figure 1.

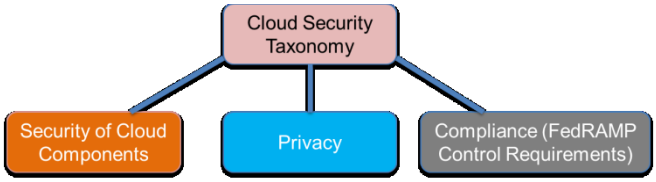

Figure 1. Cloud security taxonomy used for identifying the security controls.

Securing cloud systems involve securing the infrastructure, network, hosts, applications and data through a wide range of mechanisms such as Identity, Credentials and Access Management (ICAM), Data Segregation and Regulatory Compliance (Figure 2).

Please note that we only consider the security of the VMs in this work rather than the underlying physical hosts.

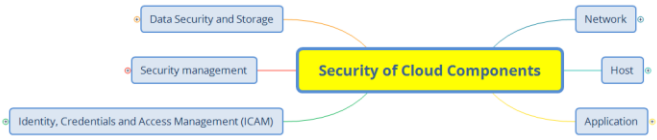

Figure 2. Cloud components security taxonomy.

One of the main goals of securing the cloud is ensuring users' data privacy. Figure 3 depicts a breakdown of data

privacy concerns and governing principals which can be used to define the users' privacy requirements.

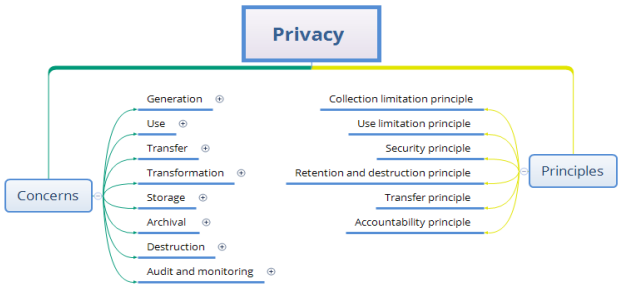

Figure 3. Privacy of the cloud can be described by the concerns and principles.

For these systems to be secured, cloud service customers, be it from the government or the private sector, need to make sure the cloud service providers satisfy a given set of security requirements. This is where FedRAMP comes into play as it assists government agencies in meeting the mandated FISMA requirements for cloud systems, and may be used by cloud service customers from the private sector as guidance when implementing their cloud security requirements. This paper does not intend to discuss the specifics or FedRAMP but rather use its security controls based on the NIST SP 800-53. The categorization (Low, Moderate, High) of the system at hand is done through FIPS PUB 199. Then the set of security controls corresponding to the baseline need to be implemented. The security controls can be grouped into three categories: Technical, Operational, and Management. In this paper, we do not address all the security controls but only the technical ones which need to be implemented on the VMs.

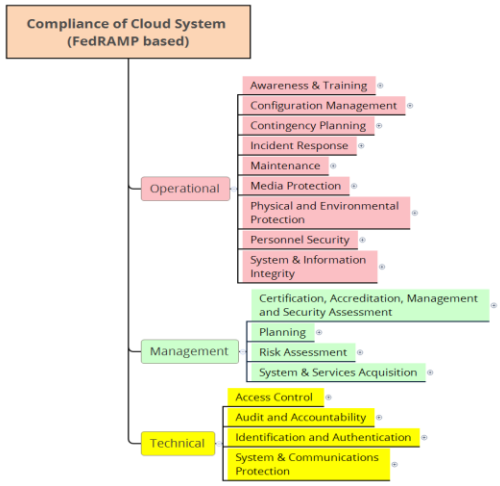

Figure 4. For the compliance, FedRAMP based taxonomy [29] can be used.

IV. Cloud Security Management Automation

In our cloud security management approach, we focus on the security automation for the Infrastructure-as-a-Service (IaaS) services offered by CSPs where users create, operate, and manage VMs with the requested virtual resources (e.g., number of cores, amount of memory, storage requirements, co-processors such as GPU) with full access in most cases using a dashboard since manual security management of the environment is improper.

The proposed cloud security management approach is shown in Figure 5. In our architecture, the user specifies the security requirements of the needed cloud environment and the VM definitions (e.g., the VM required resources) through an editor. Through a communication channel (e.g., Secure Shell (SSH), or Secure Sockets Layer (SSL)), the user's requirements are passed to the configuration engine. Next, the configuration engine interfaces with the CSP to create the requested cloud environment that satisfies the user security requirements. Once the system is set-up, the configuration engine interfaces with the CSP to continuously monitor the cloud environment and to notify the users if any abnormal configuration or behavior is detected. Below, we provide further details of each step in the form of a proof of concept.

As an example, a user wants to create a cloud environment for a web application that consists of a web server and a database (DB) server (Figure 6). In this example, the user wishes to specify the ports which will be used for communications among VMs, administrators, and the users of the web application. In addition to the required VM deployment information (such as the VM name, VM size), the user also needs to implement the selected security controls to satisfy the security requirements.

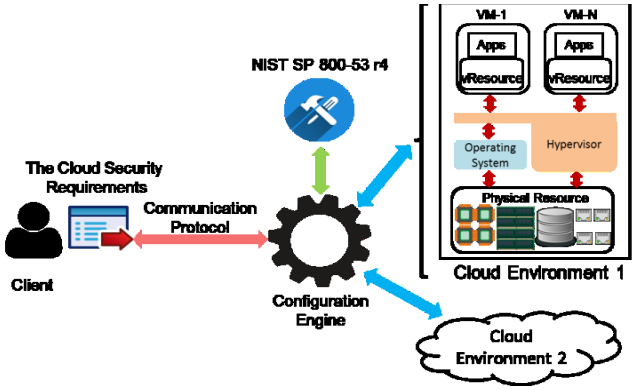

Figure 5. Cloud security management architecture.

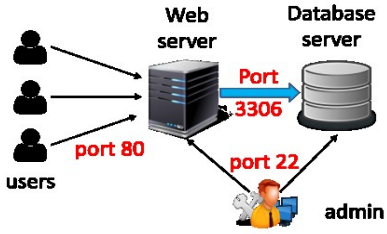

Figure 6. An example of a web application deployment.

One of the main questions that needs to be answered is how the users will specify their security requirements. NIST SP 800-53 revision 4 is a comprehensive catalog of security controls for all U.S. federal information systems except those related to national security. Therefore, we suggest using NIST SP 800-53 to implement the user requirements. Please note that not all the requirements can be met by the CSP due to lack of capabilities, which can create some gaps initially. In other scenarios, during the lifecycle of a cloud environment, the CSP capabilities may

change (and cannot provide the previous capabilities anymore). In such cases, the user may seek other providers.

For the security control selection, we define the parameters based on the taxonomy we have presented in Section III. Like TOSCA standard, we believe that a user-friendly data serialization standard with an easy to read and edit syntax should be used; therefore, we chose to encode the requirements in YAML [24].

Figure 7 depicts the YAML based definition for the given example. The user encodes the VM deployment information as well as required security controls. The user defines the VM resource requirements (flavors) as ***m1.large*** and ***m1.medium*** (1 core, 2GB memory, and 20GB storage size), respectively (line 3 and 10). Then, based on the already implemented security controls by the CSP, the configuration engine gets the lists of the remaining security controls, script name and parameters, to be applied to satisfy the user's requirements. For the given example, the user selects "***AC-2*** Account Management" that requires monitoring the use of information system accounts, "***AC-7*** Unsuccessful Logon Attempts" that limits unsuccessful logon attempts, and finally "***SC-7*** Boundary Protection Control Enhancement (5) Boundary Protection | Deny By Default / Allow By Exception" that blocks all the network ports by default unless they are specified to be open. CSP offers a security control mechanism that checks the logged in accounts to verify if the account management is being fulfilled (line 6 and line 14). To control which users are valid, the script requires a list of users (comma separated) in this scenario. Hence, the user defines that the web server needs to have only ***user1*** and ***user2*** logged in to the system and any other user would be considered as a malicious user. Similarly, only ***user3*** is accepted for the DB server.

```
1.  VM:
2.    - name: web_server
3.      flavor: m1.large
4.      controls:
5.        - script: check_current_users
6.          parameters: "-l user1,user2"
7.        - script: check_login_attempts
8.          parameters: "-w 3 –c 5"
9.    - name: db_server
10.     flavor: m1.medium
11.     controls:
12.       - script: check_open_ports
13.         parameters: "-p 22,3306"
14.       - script: check_current_users
15.         parameters: "-l user3"
```

Figure 7. VM environment definition given to the configuration engine.

In addition, since DB server will be acting in the background and does not need to interact with the end-users, the admin can whitelist port 22 and 3306 (lines 12 and 13) so that the use of any other port will be flagged as an illegal action. Finally, for the security control ***AC-7*** to check unsuccessful logon attempts, the CSP requires a warning and a critical level (line 7). When the unsuccessful login attempts exceed the given number of warnings, the user is notified. When the number of attempts is beyond the critical level, the remote connection can be temporarily blocked as a security measure mitigating a brute force attack or a dictionary attacks carried on the VMs.

Please note that each security control is an individual script based on the user requirements and CSP capabilities that operate as an agent on VMs.

The configuration engine is responsible for creating the requested VMs (with the given resources requirements), applying the specified security controls, and monitoring the VMs continuously to see if there are any requirements which are not met and notify the users of the current state of the environment. In Figure 8, we present the configuration engine algorithm. In this approach, the configuration engine reads the configuration file given in Figure 8 (line 1) and creates the requested VMs with the given resource set and VM names (line 2). This results in a list of created VMs with their IPs, called VM_list. In order to upload the security controls implementation scripts (line 4) and configure the VMs (line 5), the configuration engine waits until the VMs are active and accessible. The security controls the CSP is offering are uploaded to the VMs based on the requirements and then the VMs are configured accordingly to operate. Next, the monitoring control center (the monitoring capabilities of the configuration engine) is configured so that the created VMs can be monitored and their states are logged to a database (line 6). And, during their lifecycles, the VMs are monitored continuously (line 7). If there is any state that does not meet the requirements (for example, if any user other than user1 and user2 exist on the web_server), the users are notified so that they act accordingly.

```
1.  configuration ← read_yaml
2.  VM_list = create _VMs(configuration["VM"])
3.  wait_VMs_active(VM_list)
4.  upload_security_scripts(VM_list,
    configuration["VM"])
5.  configure_VMs(VM_list, configuration["VM"])
6.  configure_monitoring_control_center(VM_list)
7.  while(true):
8.      states = monitor_VMs(VM_list,
    configuration["VM"])
9.      If(states not expected):
10.         notify_user(states)
```

Figure 8. Configuration engine algorithm.

Furthermore, during the lifecycle of the VMs, the user's security requirements may change. Thus, the configuration engine should also allow updating the security controls.

V. Proof of Concept Implementation

In this section, we present the proof of concept implementation environment, the tools used, and how they are configured and automated. For the implementation and testing, we have created an OpenStack based private cloud environment. OpenStack is an open-source cloud stack for building public/private clouds using multiple homogenous and heterogeneous systems and managing large pools of compute, storage, and networking resources through a dashboard or using

APIs [31]. Figure 9 shows the topology of the testbed to experiment with and evaluate the proposed methodology. Our OpenStack environment consists of one separate controller node (which manages the cloud environment) and multiple compute nodes (enabling VM operations using a hypervisor). The controller node has the required OpenStack services such as Glance (image service) and Keystone (the identity manager) as well as OpenStack Python libraries for the automation (which is shown as management middleware). For the compute nodes, we have allocated three Dell XPS 8700 towers with i7 4770 processors and 12GB memory, running Ubuntu 16.04 Server for the hosts operating systems. For the virtualization, we have chosen Kernel-based Virtual Machine (KVM) hypervisor as it is supported by the Linux kernel. For the communication of the OpenStack services, an internal network switch is used.

For the continuous monitoring of the cloud environment, Nagios 3 [32] has been chosen as it is the go-to tool for remote system monitoring with flexible agent support. We have installed and configured Nagios on an individual VM and integrated with NDOUtils to provide database support which will be used to query the current and previous states of the VMs. As shown in the previous section example, multiple agents have been created based on the NIST SP 800-53 using Nagios NRPE. NRPE allows us to create custom scripts for the VMs' security controls that can be monitored by Nagios server.

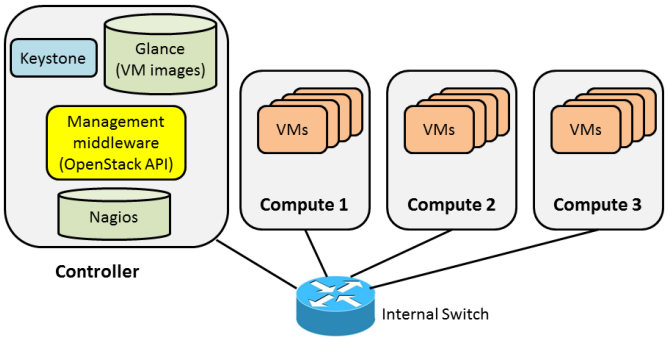

Figure 9. Proof of concept testbed architecture.

In Figure 10, we present a breakdown of the configuration time. The engine spends most of its configuration time, 33 seconds, creating the VMs and waiting for them to be accessible (i.e., active) while uploading the scripts and configuring the VMs (and Nagios server) take 2 seconds each. After the VMs are configured, it only takes 0.04 seconds to get the update from the Nagios DB. Not to mention that the agents used for the security controls implementations are lightweight in that their load on the system is negligible, and are checked periodically for a configurable amount of time, 5 seconds in this case; consequently, the agents do not introduce overhead which would impact the users' operations on the cloud environment.

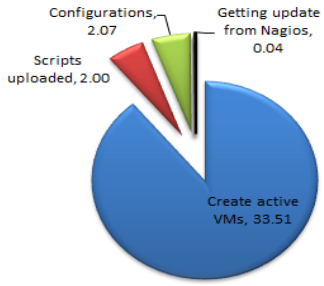

Figure 10. Time distribution of major configuration engine operations (in seconds).

Below, in Figure 11, we show an example of the user notification. In Figure 11(a), the services show that there are no unauthorized users, i.e. the logged in users are from the set of given allowed users. In Figure 11(b), the user is notified of ports that were not supposed to be open. Port 5666 is used by Nagios communication (i.e., NRPE service) and since it was not specified as an allowed port, the user is notified.

```
OK -- No unauthorized users
```

(a) Logged in users with no security violation

```
CRITICAL -- Unauthorized ports open.
Unauthorized ports: 5666
```

(b) Unauthorized open ports are notified

Figure 11. User notification example

VI. Conclusion

Even though the cloud computing systems are highly popular for personal usage, for organizations and government agencies, security of their cloud infrastructure is still a major concern. Current techniques for cloud security are manual and error prone which introduces additional vulnerabilities. Hence, it is critically important to develop a cloud security automation methodology that will be used to configure cloud systems that meet user security requirements. Thus, in this paper, we demonstrated a methodology that is based on the NIST SP 800-53 security controls and Nagios monitoring tool to implement the selected security controls using cloud service provider's capabilities. We have demonstrated a proof of concept implementation using OpenStack in a private cloud environment. As future work, we are planning on including multiple cloud systems as well as the ability to launch automated/semi-automated actions when there is a security control violation.

Acknowledgment

This work is partly supported by National Science Foundation (NSF) research project NSF CNS-1624668 and National Institute of Standards and Technology (NIST).

DISCLAIMER

Any mention of commercial products or organizations is for informational purposes only; it is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the products identified are necessarily the best available for the purpose.

The identification of any commercial product or trade name does not imply endorsement or recommendation by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose. Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

REFERENCES

[1] R. Mell, and T. Grance, "The NIST definition of cloud computing," NIST Special Publication 800-145, 2011.

[2] Report IDC FutureScape "Worldwide IT Industry, (2016), Leading Digital Transformation to Scale", New York

[3] "How Big Can AWS Get?", [Online] URL: http://wikibon.com/how-big-can-aws-get/, Accessed: June 2017

[4] "RightScale 2017 State of the Cloud Report" [Online] URL: http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2017-state-cloud-survey, Accessed: June 2017

[5] "Security Beyond the Traditional Perimeter Executive Summary", Ponemon Institute: July 2016, [Online] URL: http://cdn2.hubspot.net/hubfs/30658/Ponemon_External_Threat_2016__ExecSumm.pdf, Accessed: June 2017

[6] CSA Top Threats Working Group. "The Treacherous 12: Cloud Computing Top Threats in 2016." Cloud Security Alliance (CSA), Feb (2016).

[7] R. Chandramouli, S. Garfinkel, S. Nightingale, S. Rose, "Trustworthy Email," NIST Special Publication 800-177, Sep. 2016.

[8] B.P. Rimal, E. Choi, and I. Lumb, "A Taxonomy and Survey of Cloud Computing Systems," NCM 9, 2009, pp. 44-51.

[9] M. Schmidt, L. Baumgartner, P. Graubner, D. Bock and B. Freisleben, "Malware Detection and Kernel Rootkit Prevention in Cloud Computing Environments," 19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, 2011.

[10] C. Modi, D. Patel, B. Borisaniya, A. Patel and M. Rajarajan, "A survey on security issues and solutions at different layers of Cloud computing," The Journal of Supercomputing, pp. 1-32, 2012

[11] M. Abbasy and B. Shanmugam, "Enabling Data Hiding for Resource Sharing in Cloud Computing Environments Based on DNA Sequences," IEEE World Congress, 2011.

[12] J. Feng, Y. Chen, D. Summerville, W. Ku, and Z. Su, "Enhancing cloud storage security against roll-back attacks with a new fair multi-party non-repudiation protocol," Consumer Communications and Networking Conference, 2011.

[13] Q. Nguyen and A. Sood, "Designing SCIT architecture pattern in a Cloud-based environment," IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops, 2011.

[14] P. Zech, "Risk-Based Security Testing in Cloud Computing Environments," IEEE Fourth International Conference on Software Testing, Verification and Validation, 2011.

[15] L. Kaufman, "Data security in the world of cloud computing," IEEE Security and Privacy Journal, vol. 7, no. 4, pp. 61-64, 2009

[16] F. Heylighen, and C. Gershenson, "The Meaning of Self-organization in Computing," IEEE Intelligent Systems, 18(4), 2003, pp. 72-75.

[17] M. Parashar, and Salim Hariri, "Autonomic computing: An overview," Unconventional Programming Paradigms, 2005, pp. 97-97.

[18] H. Takabi, J. BD Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," IEEE Security & Privacy 8, no. 6, 2010, pp. 24-31.

[19] C. Tunc, F. Fargo, Y. Al-Nashif, S. Hariri, and J. Hughes, "Autonomic Resilient Cloud Management (ARCM) Design and Evaluation," International Conference on Cloud and Autonomic Computing, London, 2014, pp. 44-49.

[20] P.K. Patra, H. Singh, and G. Singh, "Fault tolerance techniques and comparative implementation in cloud computing," International Journal of Computer Applications, 2013, 64(14).

[21] M. Agarwal, V. Bhat, H. Liu, V. Matossian, V. Putty, C. Schmidt, G. Zhang, L. Zhen, M. Parashar, B. Khargharia, and S. Hariri, "Automate: Enabling autonomic applications on the grid," IEEE Autonomic Computing Workshop, 2003, pp. 48-57.

[22] "The Cloud Standards Customer Council" URL: http://www.cloud-council.org

[23] "Security for Cloud Computing Ten Steps to Ensure Success Version 2.0," URL: http://www.cloud-council.org/deliverables/CSCC-Security-for-Cloud-Computing-10-Steps-to-Ensure-Success.pdf

[24] "OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA)", URL: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

[25] T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, and S. Wagner, "OpenTOSCA–a runtime for TOSCA-based cloud applications," Springer International Conference on Service-Oriented Computing, 2013, pp. 692-695

[26] L. Trammell, "TOSCA Cloud Orchestration for Beginners" URL: http://cloudify.co/2015/07/21/what-is-TOSCA-cloud-application-orchestration-tutorial-cloudify.html, Accessed: June 2017

[27] OpenStack Heat-Translator, [Online] URL: https://wiki.openstack.org/wiki/Heat-Translator

[28] Security and Privacy Controls for Federal Information Systems and Organizations, NIST Special Publication 800-53 Revision 4

[29] "FedRAMP Security Controls Baseline", Accessable: https://www.fedramp.gov/resources/documents-2016/

[30] CloudFlare DDoS protection, URL: www.cloudflare.com/lp/ddos-a/

[31] "OpenStack: Open source software for creating private and public clouds", URL: https://www.openstack.org

[32] "Nagios: The Industry Standard in IT Infrasturecture Monitoring", https://www.nagios.com