Application of Quantitative Security Metrics In Cloud Computing

Research · December 2015

CITATIONS
0

READS
101

3 authors, including:

Kennedy Torkura
Hasso Plattner Institute
7 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Application of Quantitative Security Metrics In Cloud Computing

Kennedy A Torkura, Feng Cheng and Christoph Meinel
Chair of Internet Technologies and Systems
Hasso Plattner Institute, University of Potsdam
14482, Potsdam, Germany

kennedy.torkura, feng.cheng, christoph.meinel@hpi.de

Abstract—**Security issues are still prevalent in cloud computing particularly public cloud. Efforts by Cloud Service Providers to secure out-sourced resources are not sufficient to gain trust from customers. Service Level Agreements (SLAs) are currently used to guarantee security and privacy, however research into SLAs monitoring suggests levels of dissatisfaction from cloud users. Accordingly, enterprises favor private clouds such as OpenStack as they offer more control and security visibility. However, private clouds do not provide absolute security, they share some security challenges with public clouds and eliminate other challenges. Security metrics based approaches such as quantitative security assessments could be adopted to quantify security value of private and public clouds. Software quantitative security assessments provide extensive visibility into security postures and help assess whether or not security has improved or deteriorated. In this paper we focus on private cloud security using OpenStack as a case study, we conduct a quantitative assessment of OpenStack based on empirical data. Our analysis is multi-faceted, covering OpenStack major releases and services. We employ security metrics to determine the vulnerability density, vulnerability severity metrics and patching behavior. We show that OpenStack's security has improved since inception, however concerted efforts are imperative for secure deployments, particularly in production environments.**

Keywords: Cloud security, cloud vulnerabilities, vulnerability assessment, vulnerability lifecycle, security metrics.

I. INTRODUCTION

Security remains a key decision factor for migration of resources to the cloud [1], several enterprises are hesitant to adopt cloud technologies because of perceived security uncertainties. Current cloud security measures do not sufficiently prevent threats and vulnerabilities in cloud infrastructure and Cloud Service Providers (CSPs) do not quantitatively and objectively demonstrate what security guarantees they offer. In most cases, cloud customers rely on SLAs as guarantees for security, however SLAs are not convincing enough hence the need for SLAs monitoring [2]. A potential approach consists in quantitative security analysis of cloud software using specific security metrics. Security metrics offer objective assessment for measuring software vulnerabilities and risks [3]. These metrics analyze historical vulnerability data to gain insights into software vulnerability trends. The results are critical for objective decision making, secure software development, patch management and risk assessment [4] [5]. However, security metrics has not gained attention in cloud security. Luna et al [6] proposed a framework for Infrastructure as a Service (IaaS) cloud security metrics but whether this framework suits private or public cloud is yet to be clarified. The security challenges of private and public clouds are not entirely the same. While there are several similarities, differences also exist. Current trends [7] indicate a drive towards enterprise private clouds as they offer more control and security visibility. A challenge with private cloud is the shortage of open-source cloud software especially when compared with the closed-source options. Also, cloud software are yet to be properly scrutinized for security as done for traditional software such as Windows and Linux [8]. OpenStack [9] is the leading open-source cloud software [10], suiting both private and hybrid clouds. Its growth and popularity has been applauded however, its deployment in production environments is debatable. Most OpenStack deployments are either experimental, research or proof-of-concept [11]. A number of issues including security and lack of required skill sets, are responsible for this deployment state. A quantitative security assessment of OpenStack may determine its current security state and provide better security visibility [12]. However, to the best of our knowledge, there are no assessments of this nature hence our work. This paper presents an empirical security assessment of OpenStack by addressing the following:

Q1. What is the applicability of security metrics to cloud computing, would the results differ from its application in traditional computing?
Q2. What is the current security status of OpenStack, from an objective perspective?
Q3. Are there specific challenges to application of security metrics to cloud computing?
Our major contributions are as follows:

- We conduct an empirical security assessment of OpenStack releases and services.
- We identify the most prevalent vulnerabilities in the OpenStack eco-system and recommend specific areas

that require more attention from developers and also adopters.
- We show that security metrics could improve security in cloud computing.

This rest of this paper is organized as follows: Section II introduces the related works that we applied. Section III reviews OpenStack background, architecture and current security developments. We discuss on security metrics in Section IV. The actual analysis of the OpenStack vulnerabilities is in Section V while further discussion on our findings are made in Section VI. Highlights of future research directions and our conclusion are summarized in Sections VII and VIII respectively.

II. RELATED WORKS

There is little literature on OpenStack security evaluation. Fall et al [13] applied Fault Tree Analysis to evaluate the security of OpenStack. The aim of their work was to evaluate the security risks in the inter-connectedness of OpenStack services. In [14], practical experiments were conducted to identify multitenant vulnerabilities in OpenStack deployments. We cover a broader scope by analyzing OpenStack services and releases. Most previous works on quantitative security metrics were applied in traditional computing. Alhamzi et al [8] applied quantitative security assessment to major OSs. In [15], an analysis of patch development process for 10 popular client applications and over 1,000 vulnerabilities discovered several security issues. Luna et al [6] proposed a framework for applying security metrics in cloud computing. The possible challenges and benefits of the framework were presented, however there was no practical application of their recommendations. We considered some of their proposals in this paper and realized challenges not mentioned in their work. These challenges include larger attack surface owing to the requirement for constant internet access [16] and prevalence of shared vulnerabilities in cloud software because of the drive towards tight integration.

III. REVIEW OF OPENSTACK

OpenStack [9] is an open-source cloud computing software developed and maintained by an open community of developers under the auspices of the OpenStack foundation. Originally launched as a joint project of National Aeronautics and Space Administration (NASA) and RackSpace, OpenStack has tremendously grown and is currently supported by more than 200 companies. OpenStack development follows the bazaar approach to software development, and benefits from a broad range of skill-sets from over 1,500 developers worldwide. Security recently gained more interest in OpenStack with the creation of the OpenStack Security Project. This section briefly describes the overall architecture of OpenStack's latest release and highlights OpenStack's security structure.

A. *OpenStack Kilo Architecture*

OpenStack Kilo is the eleventh OpenStack release. Some features introduced with Kilo include port security for OpenVSwitch and the first release of bare-metal provisioning (Ironic). OpenStack is an ecosystem aimed at providing flexible cloud computing applications, hence it consists of several services. The main services are listed below:

1) *Swift*: provides a scalable and highly available object storage service. It is structured as an open-source alternative to Amazon Web Services (AWS) Simple Storage Service (S3) and has several APIs inter-operable with AWS S3.
2) *Keystone*: Provides identity, authentication and authorization services.
3) *Horizon*: Provides an intuitive web interface (dashboard) for users and administrators.
4) *Nova*: Fashioned for provision of compute services such as Virtual Machine (VM) provisioning.
5) *Neutron*: Neutron provides virtual networking features for other services.
6) *Cinder*: Provides software-defined block storage consumable by Nova VMs.
7) *Glance*: Provides catalog and repository service for images.

B. *OpenStack Security Structure*

The OpenStack security project [17] was recently introduced to improve the overall security structure of OpenStack ecosystem. The project consists of a Security Team and a Vulnerability Management Team (VMT). Also, some security focused software were introduced, more details on these is provided below:

1) OpenStack Security Structure: OpenStack's security team consists of core developers and security developers. The security team is responsible for the production of security notes and developer guidance documents [17]. On the other hand, the OpenStack VMT ensures timely management of vulnerabilities through the Vulnerability Management Process (VMP). The VMP starts on reception of a vulnerability report. This is followed by bug verification, initial impact assessment and a bugtask creation. Next, patch development is scheduled in the current branch and other affected ones. A patch review process follows, core developers are involved in the process onwards. Concurrently, the process of publishing a security advisory is triggered. Also, a Common Vulnerabilities and Exposures (CVE) identity is requested from National Vulnerability Database (NVD), this will later be used for vulnerability disclosure. Following approval by reviewers, patches are pushed to the respective branches and security advisories are published. The vulnerability is then officially disclosed through NVD and other appropriate channels.

2) *OpenStack Security Products*: There are basically two software development efforts focused at enhancing integral robust security in OpenStack [17]. Anchor is a lightweight, Public Key Infrastructure (PKI) for provisioning cryptographic

certificates within OpenStack services. Bandit is a security linter for Python, it can be run against Python code to identify security flaws. Bandit is suitable for secure coding practices in OpenStack, which is largely developed with Python programming language.

IV. SECURITY METRICS

A security metric is a quantifiable measurement that indicates the level of security for an attribute in a system [5]. Several security metrics have been proposed [3] in academia and industry such as vulnerability density, relative vulnerability, attack surface, severity-to-complex and security scoring vector. Security metrics haven't gained attention in cloud security [6] yet they offer benefits such as security evaluation of CSPs [18]. Luna et al [6] proposed a framework for quantitative, objective security measurements in the cloud. We consider portions of their proposal, our work analyzes OpenStack's security using vulnerability density, vulnerability lifecycle metrics, patching trends and vulnerability severity. We selected these metrics since combining more than one security metric affords better security value [5]. Also, there is available, open data for conducting the necessary analysis and our results can be easily verified by interested parties. More so, results of similar analysis of software are available for comparative analysis. In the next subsection, we introduce the security metrics we have applied in this work.

A. Vulnerability Lifecycle

Fig.1 shows the lifecycle of a vulnerability and associated risks users are exposed to within this time. Vulnerability lifecycles refer to the period from when a vulnerability is discovered in a software until when it is completely eliminated [15]. Software vulnerabilities undergo the following stages:

1) Discovery Time (t_v): The discovery time is the earliest time a vulnerability is discovered. Vulnerabilities could be discovered by software vendors, security researchers or malicious persons e.g. hackers. Most often, the discovery time is kept secret until after the vulnerability is fixed. However, third party researchers could force vendors to disclose a vulnerability even if a patch has not been provided.

2) Exploit Time (t_e): The exploit time is the time at which a tool is developed to take advantage of a software vulnerability to compromise security. Exploits are usually available in hacker communities and underground exploit markets. We do not consider this metric in this work.

3) *Disclosure Time (t_d)*: The disclosure time is the specific time the existence of a software is publicly announced. There are several channels through which such announcements are made, e.g. vendor security advisories, NVD and third-party vulnerability databases.

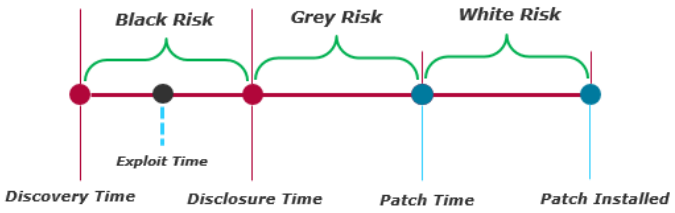

Figure 1. Vulnerability Lifecycle showing the stages of a vulnerability from injection to elimination.

4) Patch Time (t_p): The patch time refers to the time when software vendors provide suitable patches to re-mediate existing security vulnerabilities. The patch time could be before, at or after the disclosure time. Third parties also provide patches occasionally but these are not always trustworthy and could be malicious. In this work we consider only the official patches acquired from OpenStack Security Advisories (OSSA).

B. Risks Involved in Vulnerability Lifecycles

Users of vulnerable software are exposed to various security risks during the lifecycle of a vulnerability [19], these risks (Fig.1) are explained below:

1) Black Risk: This is the risk between discovery time and disclosure time. During this period, limited information about the existence of a vulnerability is available except to few people who could be good-intentioned or malicious.

2) Grey Risk: Grey risk emerges between disclosure time and patch time. In this work, we analyze the grey risk for OpenStack.

3) White Risk: After the release of a vulnerability risk, software could still be exploited if not yet patched by users. The risk within this period is termed white risk.

C. Vulnerability Density

Vulnerability density is analogous to defect density in software reliability engineering. While defect density measures the number of defects per lines of code, vulnerability density measures the amount of vulnerabilities per lines of code. Vulnerability density is suitable for comparing versions of a software and software of similar families. Vulnerability density has been applied to several software including OSs such as Microsoft Windows and RedHat Linux [20] and web servers: Apache web server and Microsoft Internet Information Services (IIS) [21]. Measurements derived in these works gave useful insights into software security. Vulnerability density metric is suitable for risk assessment, decision-making during software testing and maintenance planning [21]. It demonstrates the level of secure coding practices implemented during the development of a software. Vulnerability density is derived from the following formula [21]:

$$VD = \frac{no\ of\ vulnerabilities}{ksloc} \quad - \quad (1)$$

where VD represents the Vulnerability Density, no of vulnerabilities is the total number of vulnerabilities discovered in the software under test and ksloc is the size of the code in kilo lines of code (ksloc).

V. ANALYSIS OF OPENSTACK VULNERABILITES

In this section we analyze OpenStack, we aim at measuring its security based on historical data to identify the prevalent vulnerabilities. We explain our approach and examine the results we obtained. We believe our results would be useful for further research and enterprise deployments.

A. Data Collection

Results from security analysis of historical data provide empirical characteristics of software vulnerabilities [3]. Such analysis are useful for software security assessments and security prediction [22]. To gain insights into the vulnerability trends of OpenStack, we collected vulnerability data from the NVD [23] and Open Source Vulnerability Database (OSVDB) [24]. The NVD is a public database of security information managed by the U.S. National Institute of Standards and Technology (NIST), while OSVDB is a third party vulnerability repository. These data sources are commonly used for security analysis hence our choice. We correlated data from both sources in an effort to improve accuracy, since previous researchers have reported about inaccuracies in these sources [15] [25].

B. Measuring OpenStack Known Vulnerabilities

OpenStack software development consists of major release cycles, within these cycles versions are also released periodically. However, in this work we focus on the major releases for the sake of simplicity. OpenStack release cycle is bi-annual, followed by 3 - 6 versions releases before the succeeding major release [26]. Hence adopting a version-based analysis could be complex and produce results not very different from what we obtained. We did not find records of the vulnerabilities for the first three releases: Austin, Bexar and Cactus, hence these releases are not analyzed here. Following the collation of vulnerabilities from the vulnerability sources earlier introduced, we discovered 194 vulnerabilities affected OpenStack since inception (Fig.2). We realized that 45.64 % of the total vulnerabilities are shared vulnerabilities, inherited from previous releases. There was a rise in both pure and shared vulnerabilities up till Folsom, which was most affected with vulnerabilities. Folsom had 47 vulnerabilities, while Kilo has 6 so far. However, a downward trend of vulnerabilities is observed after Folsom, this indicates an improvement in OpenStack security.

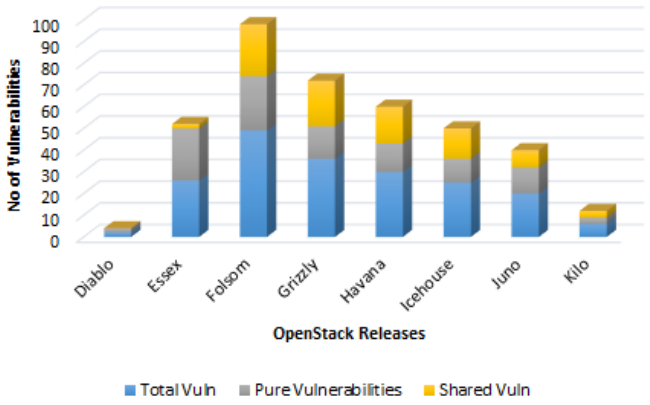

Figure 2. Histogram of OpenStack Vulnerabilities from Diablo to Kilo Releases.

C. Measurement of Vulnerability Density

Vulnerability density helps in evaluating the security value of a software from a vulnerability standpoint. We applied this metric to determine if secure coding practices have been implemented in OpenStack. The sizes of OpenStack releases were obtained from Stackalytics [27], an OpenStack project dedicated to provision of transparent and meaningful statistics for all OpenStack projects. Software sizes are expressed either in kilobytes or in Kilo Source Lines of Code (KSLOC). We decided to adopt the latter owing to its wider usage both in research and industry [5] [21]. In Table I, we show the detailed results of vulnerability density for OpenStack releases. The results are similar with those shown in Fig.2, however the trends contradicts similar measurements in [21] which asserted that the vulnerability density increased with code size. Icehouse has a better vulnerability density value than Diablo despite having a code size more than thrice Diablo's size. The result for vulnerability density metric indicate improved security.

D. Vulnerability Severity

Table II is an overview of Common Vulnerability Scoring System (CVSS) base scores for OpenStack Diablo to Juno. The scores were extracted from data obtained from NVD. CVSS severity metrics are useful for understanding severity, impact

TABLE I. VULNERABILITY DENSITY OF OPENSTACK RELEASES

Releases	KSLOC	Known Vulnerabilities	VKD
Diablo	478,671	2	4.182e-6
Essex	558,368	26	4.656e-5
Folsom	667,895	49	7.336e-5
Grizzly	1,323,479	36	2.720e-5
Havana	1,729,137	30	1.735e-5
Icehouse	1,766,546	25	1.415e-5

TABLE II. OVERVIEW OF OPENSTACK CVSS BASE SCORES

Release	Highest Score	Lowest Score	Median of Severity
Diablo	5.5	4.3	4.9
Essex	7.5	2.1	4
Folsom	7.5	1.9	4.65
Grizzly	7.1	1.9	4.3
Havana	7.5	1.9	4.15
Icehouse	7.1	2.6	4.3
Juno	6.8	3.5	4.3

and exploitability of vulnerabilities [28]. The CVSS is the "defacto" standard for measuring severity of vulnerabilities. It consists of a base metric, temporal metric and environmental metric. The base metric is the most used of all the severity metrics. We calculated the overall severity of OpenStack by first computing the median of CVSS scores for all vulnerabilities across OpenStack releases. Thereafter, the median of medians was computed (4.3). We observe that the median of severity for the releases climaxed at Folsom and progressively decreased until Juno. Grizzly, Icehouse and Juno score lowest at 4.3, while Essex, Folsom and Havana are highest with 7.5 each.

E. *Patching Behaviour*

We studied OpenStack patching behavior to determine if patches for vulnerabilities are timely provided. The aim was to identify when patches were provided and if there were unpatched vulnerabilities. To achieve this, we obtained data on vulnerability disclosure dates from NVD. However, data available at NVD does not include patch release dates therefore we referred to OSSA [29]. Fig.3 shows the aggregated patch data for each OpenStack release as at 25th September, 2015. We observed that 64.95 % of vulnerabilities were patched before disclosure, while 32.95 % are patched after disclosure date. The percentage of vulnerabilities patched on disclosure date was 2.06 %. Generally, the results of patching behavior analysis are in line with [12] which asserts that the security of open source software is more efficient at vulnerability fixing than their closed counterparts. We also realized that the patching rate of OpenStack is faster than most software products as shown in [30] e.g. RedHat Linux. In comparison with Microsoft, Apple and Oracle, which patch 70 % of their vulnerabilities before or on disclosure date [30], OpenStack's performance is commendable. Reasons for the quick patching time could be rich experience from the contributing partners, e.g. we found some patches at Red Hat Security advisories. There are no unpatched vulnerabilities discovered, this is also a positive indicator for OpenStack.

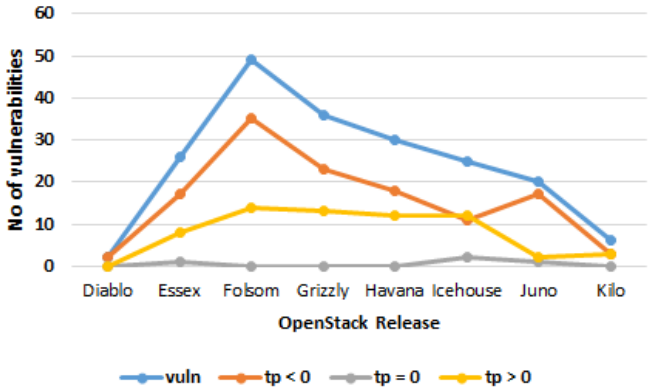

Figure 3. Patching trends across OpenStack releases showing patches released before (tp<0), on (tp= 0) and after (tp>0) vulnerability disclosure.

F. *Vulnerability Analysis of OpenStack Major Services*

We analyzed vulnerability trends for the major OpenStack services to identify the most vulnerable services. Based on empirical data from NVD, we plot a histogram (Fig.4) and show that the most vulnerable services are keystone, horizon and glance. These are the most demanded services [31], Woo et al [21] posited that market factors such as demand affect software vulnerability discovery process i.e. higher demand linearly correlates with vulnerability discovery.

VI. DISCUSSION

In this section, we discuss the possible reasons for the results we obtained from our measurements. We observe that OpenStack vulnerabilities climaxed at Folsom and dropped in subsequent releases. Interestingly, almost half of these vulnerabilities were inherited from previous releases, three consecutive releases were affected in some cases. For example, CVE-2013-2161 was the most severe vulnerability, having a CVSS score of 7.5. This vulnerability affected three consecutive releases; Folsom, Grizzly and Havana. Alhazmi et al [32] showed the implications of this trend, caused when parts of a software are reused in succeeding versions and releases. This trend could be prevalent in other cloud software because the drive towards tightly integrated services [33]. Nappa et al [15] demonstrated the risks of code sharing prevalent in Mozilla products e.g. Thunderbird and Firefox since they share similar code base. In this case, vulnerabilities patched in one product could still be exploited in another product. Similarly, malicious hackers could take advantage of this trend to develop zero-day exploits in new software version releases. OpenStack could be exposed to these risks, however further investigations that apply exploit specific security metrics could provide better insights. We observe a linear correlation between the total number of

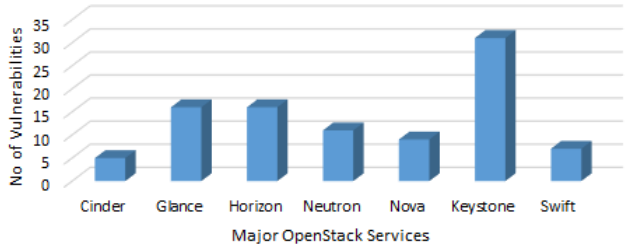

Figure 4. A histogram showing spread of vulnerabilities across OpenStack major services.

vulnerabilities and vulnerability density. This contradicts earlier assertion made in [32] which claimed that vulnerability metric worsens as code size increases. This difference could be attributed to better secure coding practices and use of Python programming language [34] in OpenStack development. Python is favored in the information security community owing to its inherent security qualities [35]. For OpenStack services, our analysis showed that keystone, horizon and glance are the most vulnerable services. The high demand for these services could be a reason for this result [21]. Keystone is the only compulsory service for deploying OpenStack, other services are optional, and hence keystone presents severe risks. Integrating third party authentication and authorization protocols such as OAuth and Multi Factor Authentication (MFA) could be a safer deployment approach. This approach is currently supported. Vulnerabilities in horizon and glance could be mitigated by applying Cross-Site Scripting counter-measures, QEMU hardening and other measures contained in OpenStack security guide.

Threats to Validity: We had to manually cross-check data collected from NVD and OSVDB for errors and omissions. Also we acquired data for discovery date and patch time by manually checking through OSSA and their Launchpad bug repository. Our case is not unique, most researchers have expressed similar challenges for quantitative assessments using historical data.

VII. FUTURE WORK

An analysis of specific vulnerabilities prevalent in OpenStack was not the focus of this work. However, in-depth analysis of the most vulnerable services: keystone, horizon and glance could provide useful insights. Similarly, an investigation into effects of programming languages on software security is an interesting research direction. Furthermore, a comparative study on several open-source and closed-source cloud computing software such as AWS and Eucalyptus, using quantitative security metrics could be interesting. The results of such analyses could be applied in selection of cloud services based on security requirements, already there are existing approaches focusing on other requirements such as performance and SLAs.

VIII. CONCLUSION

In this paper, we applied quantitative security assessment approach to cloud computing using OpenStack as a case study. We derived empirical data from NVD, OSVDB and OSSA. Our analysis shows that 194 vulnerabilities have affected OpenStack since inception. While patches were provided for all vulnerabilities, patching behavior analysis shows that 64.95% of vulnerabilities disclosed are patched before the disclosure date, 2.06 % on the disclosure date (0-day patch) and 32.99 % after disclosure. Similarly, OpenStack's average CVSS was 4.3, a medium severity rating. We also observed that the number of discovered security vulnerabilities peaked in Folsom release but experienced a downward trend thereafter. Also, our analysis suggests OpenStack keystone is the most vulnerable service, followed by horizon and glance. Hence, appropriate countermeasures are imperative for deploying these services. It is interesting to note also that some vulnerabilities appeared in more than two consecutive OpenStack releases, a clear case of vulnerabilities owing to code-sharing. This is a trend that could be prevalent in cloud software considering the focus on tight integration. The results of the vulnerability density metric correlates with the overall vulnerability trend, contrary to previous works which suggest that vulnerability destiny worsens as code base expands. We opine this to improved secure coding practices, more thorough code-reviews and use of Python programming language for development.

REFERENCES

[1] S. Bleikertz, M. Schunter, C. W. Probst, D. Pendarakis, and K. Eriksson, "Security audits of multi-tier virtual infrastructures in public infrastructure clouds," in Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop, ser. CCSW '10. New York, NY, USA: ACM, 2010, pp. 93–102. [Online]. Available: http://doi.acm.org/10.1145/1866835.1866853

[2] P. Patel, A. H. Ranabahu, and A. P. Sheth, "Service level agreement in cloud computing," 2009.

[3] W. Jansen, Directions in security metrics research. DIANE Publishing, 2010.

[4] V. H. Nguyen and F. Massacci, "A Systematically Empirical Evaluation Of Vulnerability Discovery Models: A Study On Browsers' Vulnerabilities," arXiv preprint arXiv:1306.2476, 2013.

[5] A. A. Younis and Y. K. Malaiya, "Relationship between attack surface and vulnerability density: A case study on apache http server," in Proceedings on the International Conference on Internet Computing (ICOMP). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012, p. 1.

[6] J. Luna, H. Ghani, D. Germanus, and N. Suri, "A security metrics framework for the cloud," in Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on, July 2011, pp. 245–250.

[7] J. Frearson. 47 Per cent of EU Firms Prefer Private Clouds [Online]. Available: http://business-reporter.co.uk/2015/07/15/47-per-cent-of-eu-firms-prefer-private-cloud-services/ (Access date: 9 December, 2015).

[8] O. H. Alhazmi and Y. K. Malaiya, "Application of Vulnerability Discovery Models To Major Operating Systems," Reliability, IEEE Transactions on, vol. 57, no. 1, pp. 14–22, 2008.

[9] OpenStack Cloud Software, Internet: http://www.openstack. org/ (Access date: 9 December, 2015).

[10] Linux Operating System, Internet: https://www.linux.com/news/enterprise/ , (Access date: 9 December, 2015). Cloud -computing/784573-the-top-open-source-cloud-projects-of-2014

[11] OpenStack Superuser, "OpenStack Users Share How Their Deployments Stack Up" Internet: http://superuser.openstack.org/articles/openstack-users-share-how-their-deployments-stack-

up?utm%20content=15371724&utm%20medium=social&utm%20source=twitter, (Access date: 9 December, 2015).
[12] J. R. Jones, "Estimating software vulnerabilities," IEEE Security &Privacy, vol. 5, no. 4, pp. 28–32, 2007.
[13] D. Fall, T. Okuda, Y. Kadobayashi, and S. Yamaguchi, "Towards a vulnerability tree security evaluation of openstack's logical architecture," in Proceedings of the 7th International Conference on Trust and Trustworthy Computing - Volume 8564. New York, NY, USA: Springer-Verlag New York, Inc., 2014, pp. 127–142. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08593-7 9
[14] M. Gusev, S. Ristov, and A. Donevski, "Security vulnerabilities from inside and outside the eucalyptus cloud," in Proceedings of the 6th Balkan Conference in Informatics, ser. BCI '13. New York, NY, USA: ACM, 2013, pp. 95–101. [Online]. Available: http://doi.acm.org/10.1145/2490257.2490285
[15] A. Nappa, R. Johnson, L. Bilge, J. Caballero, and T. Dumitras, "The attack of the clones: A study of the impact of shared code on vulnerability patching," 2015.
[16] F. A. Locati, OpenStack Cloud Security. Packt Publishing, 2015.
[17] OpenStack Foundation "Security", Internet: https://wiki.openstack.org/wiki/Security , (Access date: 9 December, 2015).
[18] M. I. Tariq, "Towards information security metrics framework for cloud computing," International Journal of Cloud Computing and Services Science (IJ-CLOSER), vol. 1, no. 4, pp. 209–217, 2012.
[19] S. Frei, M. May, U. Fiedler, and B. Plattner, "Large-scale vulnerability analysis," in Proceedings of the 2006 SIGCOMM workshop on Largescale attack defense. ACM, 2006, pp. 131–138.
[20] O. H. Alhazmi, Y. K. Malaiya, and I. Ray, "Measuring, Analyzing And Predicting Security Vulnerabilities In Software Systems," Computers & Security, vol. 26, no. 3, pp. 219–228, 2007.
[21] S.-W. Woo, H. Joh, O. H. Alhazmi, and Y. K. Malaiya, "Modeling Vulnerability Discovery Process In Apache And IIS HTTP Servers," Computers & Security, vol. 30, no. 1, pp. 50–62, 2011.
[22] S. Neuhaus, T. Zimmermann, C. Holler, and A. Zeller, "Predicting Vulnerable Software Components," in Proceedings of the 14th ACM Conference on Computer and communications security. ACM, 2007, pp. 529–540.
[23] National Vulnerability Database, Internet: http://nvd.nist.gov/ , (Access date: 9 December, 2015).
[24] Open Source Vulnerability Database, Internet: http://osvdb.org/ , (Access date: 9 December, 2015).
[25] G. Schryen, "Is open source security a myth?" Communications of the ACM, vol. 54, no. 5, pp. 130–140, 2011.
[26] OpenStack Foundation, "OpenStack Releases", Internet: http://docs.openstack.org/releases/ , (Access date: 9 December, 2015).
[27] OpenStack Foundation, "Stackalytics", Internet: https://wiki.openstack.org/wiki/Stackalytics , (Access date: 9 December, 2015).
[28] H. Ghani, J. Luna, and N. Suri, "Quantitative Assessment Of Software Vulnerabilities Based On Economic-Driven Security Metrics," In Risks And Security Of Internet And Systems (CRiSIS), 2013 International Conference on. IEEE, 2013, pp. 1–8.
[29 OpenStack Foundation, "OpenStack Security Advisories", Internet: https://security.openstack.org/ossalist.html , (Access date: 9 December, 2015).
[30] M. Shahzad, M. Z. Shafiq, and A. X. Liu, "A large scale exploratory analysis of software vulnerability life cycles," in Proceedings of the 34th International Conference on Software Engineering. IEEE Press, 2012, pp. 771–781.
[31] OpenStack Superuser, "OpenStack User Survey Insights – November 2014", Internet: http://superuser.openstack.org/articles/openstack-user-survey-insights-november-2014 , (Access date: 9 December, 2015).
[32] O. H. Alhazmi and Y. K. Malaiya, "Modeling the vulnerability discovery process," in Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on. IEEE, 2005, pp. 10–pp.
[33] F. Tech, "6 ways cloud integration leads to better business performance." [Online]. Available: http://www.forbes.com/sites/oracle/2015/09/22/6-ways-cloud-integration-leads-to-better-business-performance/
[34] Python Software, Internet: https://www.python.org/ , (Access date: 9 December, 2015).
[35] S. M. Kerner, "How Open Source Python Drives OpenStack Cloud (Video)", Internet: http://www.datamation.com/cloud-computing/how-open-source-python-drives-the-openstack-cloud-video.html , (Access date: 9 December, 2015).