SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2017, Vol. 60 122102:1–122102:13
doi: 10.1007/s11432-016-0322-7

Saudi cloud infrastructure: a security analysis

Wahid RAJEH[1,2], Hai JIN[1*] & Deqing ZOU[1]

[1]*Services Computing Technology and System Lab, Big Data Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074*, China;*
[2]*School of Computers and Information Technology, University of Tabuk, Tabuk* 71491*, Saudi Arabia*

Received September 18, 2016; accepted November 17, 2016; published online April 1, 2017

Abstract The growing demand and dependence upon cloud services have garnered an increasing level of threat to user data and security. Some of such critical web and cloud platforms have become constant targets for persistent malicious attacks that attempt to breach security protocol and access user data and information in an unauthorized manner. While some of such security compromises may result from insider data and access leaks, a substantial proportion continues to remain attributed to security flaws that may exist within the core web technologies with which such critical infrastructure and services are developed. This paper explores the direct impact and significance of security in the Software Development Life Cycle (SDLC) through a case study that covers some 70 public domain web and cloud platforms within Saudi Arabia. Additionally, the major sources of security vulnerabilities within the target platforms as well as the major factors that drive and influence them are presented and discussed through experimental evaluation. The paper reports some of the core sources of security flaws within such critical infrastructure by implementation with automated security auditing and manual static code analysis. The work also proposes some effective approaches, both automated and manual, through which security can be ensured through-out the SDLC and safeguard user data integrity within the cloud.

Keywords cloud security, vulnerability detection, web security, Saudi infrastructure, cloud service

Citation Rajeh W, Jin H, Zou D Q. Saudi cloud infrastructure: a security analysis. Sci China Inf Sci, 2017, 60(12): 122102, doi: 10.1007/s11432-016-0322-7

1 Introduction

The advent of the Internet many decades ago has perhaps been one of the most influential elements that have impacted the lives of many. Since its emergence, the Internet has steadily grown in both scope and complexity. The technology has been widely adopted for various services including but not limited to Ecommerce, Voice over IP (VoIP), online gaming, streaming services and cloud services. As the Internet continues to grow and get adopted for various services, so do the associated technologies that are relied upon in the design and realization of these web platforms. With the growing reliance upon the on-line services, the security and integrity of the abundant data that is handled, transferred and stored within the cloud is constantly becoming the target of malicious activities that attempt to breach and gain unauthorized access for various reasons and motive. Recent years have seen a steady

* Corresponding author (email: hjin@hust.edu.cn)

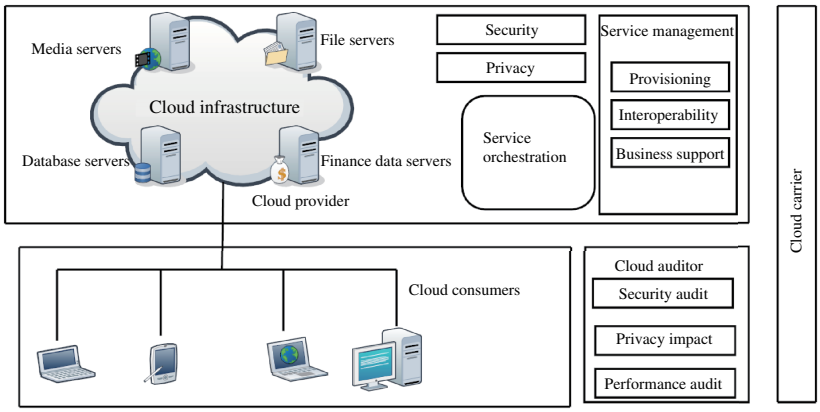

Figure 1 A generalized structure of cloud/web services.

growth in the volume and sophistication of exploits and data breaches, some of which have affected critical infrastructure and resulted in huge financial losses. Security has been and continues to remain a core concern within critical infrastructure. As more services continue to evolve and migrate to the web, making them openly accessible within the public domain, the need to safeguard and protect user data and service integrity grows accordingly. Furthermore, in an attempt to bridge the gap between citizens and governments, clients and service providers, it has become even mandatory in some countries for public services to achieve and maintain a complete web presence [1]. This trend which is gradually picking up in Saudi Arabia has seen numerous public services including education, banking, healthcare and even government, migrate to the web where such services have become more efficient in reaching the needs of citizens in even the most remote areas of the nation. With such a web presence, security analysis and awareness reports are important to ensure that the ever-growing data are safeguarded from malicious intent such as the Shamoon Trojan [2] that targeted the Saudi petroleum industry. The breach in such critical web and cloud services would not only impact on individual user privacy but could have high financial and economic ramifications on major industries. This is a core motivation of the work presented in this paper.

A majority of security concerns are related to the application layer and this is simply motivated by the fact that most X as a Service (XaaS) platforms are rendered at the application layer and accessed via web browsers amongst other application tools. The openness and availability of such services to all users have led to the steady growth of application layer attacks in comparison to other layers such as the physical and network layers. This claim is also strongly supported by current reports including the NIST database [3]. Another independent source, Gartner Group has suggested that more than 72% of the security attacks that target organizations reside within the application layer of cloud and web services, generally illustrated in the Figure 1.

Since the application layer is the final level of implementation, organizations without the man-power or development team size tend to lack the resources required to ensure proper integration of services at the application layer. Additionally, since security at the application layer is strongly dependent on security at the underlying network, internet and transport layers, critical skill is often required in ensuring that implementation at the application layer does not inherit vulnerabilities from underlying layers while not introducing its own vulnerabilities into the integrated framework that makes up a system.

The aim of this paper is precise and thus motivated. Through experimental security analysis and evaluation of a large number of platforms, the paper aims to stress methodologies, tools and practices which are crucial in ensuring application layer security of such platforms at each level of the Software Development Life Cycle (SDLC). By focusing on Saudi cloud infrastructure, the paper aims to establish a framework towards security-aware web software development as well as assuring application security standards. Moreover, since the current on-demand data products development trend is not unique to Saudi Arabia, the results and framework presented in this paper offer feasible adaptations to a majority of

middle-eastern and global scenarios. This extends the impacts and benefits of the experimental evaluation beyond the targeted country.

2 Application layer as a security focus

While web applications originally surfaced as a means for businesses and organizations to function and handle operations, the modern day web applications have become core and integral components of daily lives. The application layer remains the weakest link in the security chain and this is largely attributed to the fact that, good programming skill is simply not sufficient in guaranteeing software security at the application layer. The OWASP [4] and WASC [5] top 10 list from the year 2015 are presented below.

- Injection (SQLi).
- Broken Authentication and Session Management (BASM).
- Cross-Site Scripting (XSS).
- Insecure Direct Object Referencing (IDOR).
- Security Misconfiguration (SM).
- Sensitive Data Exposure (SDE).
- Missing function Level Access Control (MFLAC).
- Cross Site Request Forgery (CSRF).
- Use of components with known vulnerabilities (UCKV).
- Unvalidated redirects and forwards (URF).

Unfortunately, counter intuitively, organizations continue to adopt a reactive rather than proactive stance in application layer security, a trend that is further demonstrated by the experimental results presented in this paper. A core drawback with the reactive stance to security is that, in Saudi Arabia as well as in most nations where everyday lives continue to become more interwoven with cloud products and services, critical data continues to accrue within such services and to react to security only after a breach has occurred essentially puts user data at risk, a practice that could pose real economic and security concerns. Through the experimental evaluation presented in the paper, a more proactive and efficient framework of software security auditing at each level of the SDLC is encouraged for cloud service providers.

To the best of our knowledge, there has been very limited effort in the documentation of such large-scale security auditing work and when such work has been presented, data is extracted through experimental simulations that are designed and realized in controlled laboratory environments. The work in this paper on the contrary covers 70 live infrastructures that are online and accessible to the general public. The remainder of the paper is thus organized. The research background pertaining to the work presented in this paper is presented in the Section 3. The experimental security evaluation scheme is put forward and described in the Section 4, which is followed by the Section 5 where the experimentally derived data is presented and analyzed. Preventive and mitigation schemes for web and cloud services are presented in the latter parts of this section. The paper concludes in Section 5.

3 Research background

The rapid growth of the Internet and the its accompanying XaaS solutions has drawn attention from researchers from various fields with application layer security being at the heart of concerns in recent years. In the area of cloud security, there are also various sub areas that literature has focused on. While some literature has addressed the issue of cloud security in a generalized manner [6], others have attempted to examine emerging security challenges that accompany cloud infrastructure development [7]. Along the lines of implementation, other literatures have proposed recommendations and safe practices towards cloud enterprise security [8, 9]. While such work is effective in offering insight and guidelines on how safe implementation can be achieved, they fail to address true modern-day implementation trends since their recommendations are not based upon any real experimental data but rather follow general theoretical

assumptions. The work in this paper however draws from real platform implementation trends that are acquired through experimental security audits via automated tools and static code analyses that cover numerous real-time systems. In [10], the Waler tool is proposed and presented as a tool with the capability of guaranteeing vulnerability reports for each detected bug which may contain violated invariant paths. The tool offers developers the ability to effectively analyze web applications from a security point of view. This tool however fails to accurately identify software flaws that are related to FOR loops, SQL queries and regex forms. Additionally the tool's capability is only limited to servlet-based web applications. Another work in the area of tool development is presented in [11].

The tool NetCheck is designed to target the network layer in an attempt to detect vulnerabilities that have transcended from the application layer. Key capabilities of the tool are associated with its independence from application and network specifications as well as its documented ability to detect over 95% of network bug trackers within Apache, Ruby and Python. These automated tools however fail to address application layer software security vulnerabilities. An additional drawback of the NetChecker tool lies in its inability to detect application-related faults in implementation scenarios where non-socket IPC is adopted. In the domain of tool implementation and evaluation, the work in [12] presents a review of eight well known vulnerability scanners and evaluates their performances against common web application vulnerabilities in order to verify their accuracies in relation to each vulnerability. The evaluated tools were WVS, HailStorm Pro, WebInspect, Rational AppScan, McAfee SECURE, QA Edition, QualysGuard PCI and NeXpose.

The literature reported that although a majority of the tools were effective against XSS type 1 and first-order SQLi, second-order SQLi vulnerability classification proved challenging due to complicated forms of the vulnerability that are associated with other forms of keywords. The review established that Black-box scanners offered a useful means by which security auditing could be conducted at lower time and financial costs compared with static and manual code analyses. The work in this paper however effectively combines both automated tools and static code analysis into a single integrated auditing framework. More related to the work presented here in this paper is the background literature that have attempted to scan services with specific tools with the goal of discovering vulnerabilities within these services and propose means by which such flaws could be avoided in the future. Examples of such work are seen in [13–15]. In [13], the authors present a case study that covers penetration testing of a single target application. This is achieved by means of tools and static code analysis. In [14], instead of targeting a single platform and auditing it for multiple vulnerabilities, a single vulnerability, SQLi, is selected and a tool is developed towards its detection. A case study is presented towards an evaluation of the security achieved by multiple open source CMS. An analysis of some key challenges associated with cloud-aware security implementation is presented in [16]. Closely related to [17], the work provides some specifications towards the provision of more efficient security schemes within cloud systems. All such literature have reported interesting data but while the web continues to evolve in nature and complexity, so does its security state-of-the-art and the research area merits further and continuous effort. Additionally, while such efforts may effectively target specific CMS systems, they fail to cover other web development technologies.

4 Experimental methodology

Through real-time security auditing with automated tools and manual static code analysis, the paper addresses application layer security in practical and experimental manner. Contrary to previous work that target a single platform within controlled experimental environments, the paper selects 70 public and critical domain web services in Saudi Arabia towards the realization of a case study that covers numerous web application technologies with numerous real-time implementation schemes. Additionally, the target platforms are scanned over a period of 3 months, allowing for their application layer conditions to be captured under varying traffic conditions. While the paper does not claim novelty in the manner in which these automated scans and static code analyses are conducted, the work presents one of the largest scale and most diversified real-time web application security data captured on live cloud platforms.

Furthermore, while such experiments have been conducted for different scenarios in the past, the data they report is out of date as of the time of this paper and the platforms they cover are much smaller in scale compared to the work in this paper.

Drawing from the results attained from this wide-scale, real-time security audit, the paper proceeds to propose some proactive means by which service providers can ensure software security at low implementation costs at each level of the SDLC. Contrary to some previous work that blindly propose "safe-practices" in the development of web services, this paper draws from real-time live and experimental data and therefore offers strong insight into current practical scenarios.

4.1 Criteria for target and tool selection

In order to ensure diversity and reliability of the data obtained from the targets, the selection of the targets is made to incorporate 14 targets from each of the 5 economic sectors namely:

- Primary—educational industries.
- Secondary—healthcare industries.
- Tertiary—financial industries.
- Quaternary—manufacturing industries.
- Quinary—governmental services industries.

Additionally, no two targets are selected from the same web domain in order to ensure that the data captures a large and diversified scope. Here in this paper, for the security and privacy of the platforms contained within the experimental case study, the true domain names of platforms are not referenced. Platforms running beta versions are not included in the experimental dataset and neither are platforms that have not been open to the general public for more than a period of 6 months prior to the conducting of the experiments. These measures are considered sufficient in eliminating bias in the experimental evaluation.

Since automated tool scans are a core component of the experimental evaluation scheme carried out in the paper, the selection of these tools is a crucial component in ensuring the integrity of the acquired results and strong emphasis is therefore placed on this step. The selection of the tools applied in automated platform security audit is strongly guided by the OWASP top 10 security threat list. While at the moment of writing this paper, there exists not a single tool capable of recognizing vulnerabilities and identifying them in line with the OWASP list, certain well established tools prove highly efficient in providing a reference point for judging vulnerabilities in line with the OWASP list. In realizing automated target evaluation, the three tools relied upon are as follows.

- Acunetix [18]: The Acunetix vulnerability scanner offers a cloud-based solution to web security auditing. It operates by crawling a target website and automatically scans it with an extensive vulnerability database. While the tool has the potential of detecting a wide range of vulnerabilities, it is known to be highly effective in the detection of SQLi and XSS vulnerabilities.
- Nikto [19]: The Nikto tool comes in the form of an open source Common Gateway Interface (CGI) script scanner. Core features of the tool are its evasiveness and capability to operate stealthily. This evasive feature is highly effective when working with a large set of targets since experiments progress seamlessly without setting off penetration detection routines on the target platform.
- Netsparker [20]: The highly versatile tool has the potential to sniff out vulnerabilities from multiple web scripting languages. While the tools vulnerability database is not as vast as that of Acunetix, it has the capability of reporting vulnerabilities with a high level confidence.

The parameters adopted in the operation of the various automated tools are detailed in Tables 1–3.

While automated tools have the potential of offering high speeds of evaluation, previous work [13, 15] has demonstrated that static code analysis offers a fail-safe mechanism by which false positive results can be removed from experimental results while further allowing for context to be detected in some security paradigms in a way that is presently unavailable in automated tools. Static code analysis remains an integral part of the Security Development Lifecycle (SDL) of most institutions and the research community and provides a white-box means of which code can be analyzed for security loopholes. In the experiments

Table 1 Operation parameters of Acunetix vulnerability scanner

Parameter	Details
1. Profile	Default
2. Vulnerability scanner	Enabled
	a. Site crawler
	b. Subdomain scanner
	c. Blind SQL injection
3. Additional tools	d. HTTP editor
	e. HTTP sniffer
	f. HTTP fuzzer
	g. Authentication tester

Table 2 Operation parameters of Netsparker vulnerability scanner

Parameter	Details
1. Scan policy	All security checks
2. Crawling	Enabled
3. Crawl properties	a. Find and follow new links
	b. Enable crawl and attack at the same time

Table 3 Operation parameters of Nikto vulnerability scanner

Parameter	Details
1. Evasion technique	147AB
2. Pause duration	1
3. Timeout	10
4. Configuration profile	Default

presented in the paper, static code analysis is relied upon as a follow-up mechanism for isolating and confirmation vulnerabilities that have already been qualitatively detected by the automated tools.

4.2 Experimental data acquisition

While within the scope of this paper, web services are solely evaluated at the application layer, there is the need to ensure that network layer firewalls and penetration detection systems on the target platforms do not interfere with the experiments, leading to some spurious results. This is an issue for concern in conducting such cross domain experiments because some target platforms may apply IP address screening protocols in order to limit application access only to IP addresses originating from within the Saudi Arabian domain. In addressing such IP screening and restriction mechanisms, two experimental evaluation test beds are designed in order to ensure that all target platforms are effectively scanned in an exhaustive manner. The direct scan test bed which is illustrated in Figure 2 is applied to targets where IP screening routines are absent, thereby allowing for probe packets to originate and terminate from within the China Education and Research Network (CERNET).

Such direct scans allow for the evaluation command terminal to originate and terminate probe packets. On the other hand, the indirect evaluation test bed, depicted in Figure 3 is applied to targets that present with IP screening routines. The mechanism for detecting such routines is realized through a simple ping script that pings each target with 56 bytes of data and listens for a response. If a ping timeout is received, IP screening is considered present and the indirect probe approach is applied for the target evaluation.

In indirect target evaluation scenarios, the security tools which are deposited remotely on a Saudi domain PC terminal are executed via a python script which resides on the CERNET terminal. This uses the remote Saudi terminal as an unwitting VPN host in reaching the target platform for security evaluation.

Furthermore, since multiple probe packets originating from a single IP source may be regarded as Denial of Service Attacks (DOS) by security mechanisms present on some target infrastructure, a 30-

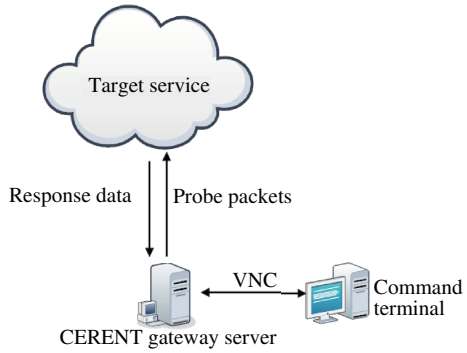

Figure 2 A direct setup toward vulnerability assessment.

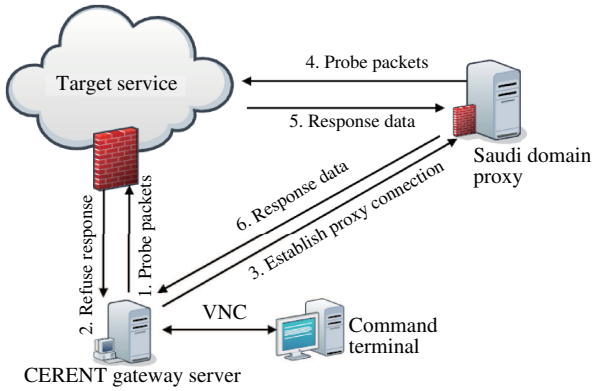

Figure 3 An indirect setup toward vulnerability assessment.

minute interval is allowed between consecutive scans in order not to trigger any anti-DOS mechanisms on the target platforms.

5 Experimental data analysis and discussion

5.1 Experimental analysis

This section presents an analysis of the data acquired through the experimental methodology illustrated in Section 4. Of all the 70 platforms targeted in this web security case study, 67 of them yielded successful and complete evaluation results while 3 targets within the academic research sector appeared to implement persistent anti-penetration schemes that significantly hindered and interfered with automated tool scans.

In line with the OWASP 2015 list, the vulnerabilities detected are sorted and presented as in Subsection 2.1. Figure 4 graphically depicts this categorization. It is crucial to point out that although the vulnerabilities are reported under these categories, the tools applied in the security evaluation do not always report the vulnerabilities with such labels. This grouping into the above categories is conducted in order to standardize the results and allow for easy comparison with previous related work. For ease of presentation, Table 4 summarizes the results by illustrating the 2 most vulnerable platforms within each sector against their vulnerability count in reference to each security vulnerability category. The vulnerability counts represent the number of independent and unrelated instances detected for each specific vulnerability.

The results depicted in Table 4 stress the criticality of the security conditions associated with the infrastructure. The infrastructure selected from the primary sector present with the highest security threat level with the target 1 being the most vulnerable of all the 70 targeted platforms. On the contrary, the lowest security threat levels are encountered within the tertiary and quaternary sectors, a result in line with preliminary analysis that suggested high security awareness within the sector as a result of

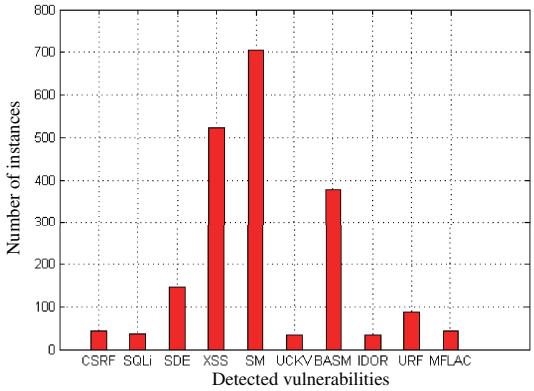

Figure 4 An overall illustration of Saudi platforms vs. vulnerabilities.

Table 4 An illustration of the target platforms and their associated vulnerability prevalence

Target		Vulnerability									
		CSRF	SQLI	SDE	XSS	SM	UCKV	BASM	IDOR	URF	MFLAC
Primary	Tgt1	–	–	102	1	55	–	6	–	–	3
	Tgt2	2	3	–	–	1	–	5	2	–	2
Secondary	Tgt1	4	1	3	–	–	–	3	1	2	–
	Tgt2	–	3	2	7	1	–	3	–	1	–
Tertiary	Tgt1	–	–	1	1	1	–	–	–	–	–
	Tgt2	1	–	3	2	1	7	–	–	6	–
Quantery	Tgt1	2	1	–	1	1	–	–	3	1	2
	Tgt2	2	–	1	2	0	1	–	–	3	6
Quinary	Tgt1	18	1	4	3	–	2	1	3	–	10
	Tgt2	2	–	–	–	5	8	–	2	1	2

the presence of penetration detection systems which hampered and in some cases, fully halted remote automated security probes.

While the most vulnerable web platform presents sensitive data exposure as the most prevalent flaw, security misconfiguration remains the most widespread and prevalent application level vulnerability amongst all platforms. In order to establish a reliable parameter in explaining the criticality of each of the application layer software vulnerabilities contained within this experimental study, prevalence p is applied and defined within the context of this paper as

$$p = \frac{\text{vulnerability instances}}{\text{number of targeted platforms}}. \tag{1}$$

Eq. (1) presents a means of handling each application layer security vulnerability in terms of p and this yields the data extrapolated in Table 4.

The quantitative data presented in Table 5 suggests that security misconfiguration is indeed the most prevalent security threat to the web and cloud platforms applied in the experimental study presented in the paper. This is a trend that further holds in Figure 4 below which illustrates the total number of experimental platforms against the overall number of detected vulnerabilities. As earlier claimed in the paper, software security is a requirement that is not completely satisfied through programming skill only. This claim is supported by the high prevalence rate attained by security misconfiguration since this security vulnerability has no direct correlation with software programming or design. We prove that through security conscious practices and routines at each level of the SDLC, such security threats could be tackled and cut down to a significant level.

Some common practical security misconfigurations encountered in software implementations of some of the most vulnerable target platforms are as follows.

- Web server session disclosure.

Table 5 The prevalence (p) of each security vulnerability across all target platforms

Vulnerability	p
CSRF	0.6
SQLi	0.5
SDE	2.1
XSS	7.6
SM	10.3
UCKV	0.5
BASM	5.6
IDOR	0.5
URF	1.3
MFLAC	0.6

- Software (PHP/ASP. NET) version disclosure.
- Outdated PHP /ASP. NET versions.
- HTTP Cookie Disclosure.
- Session cookie scoping to parent domain.
- Session token in URL.
- Deprecated SSL.

With an average prevalence of 2.9, SQLi holds the lowest prevalence of 0.5. While this may appear to attribute the SQLi security vulnerability with a low criticality, previous related work has shown that the high exploitability of the SQLi vulnerability along with the impacts of successful exploitation render it a highly critical vulnerability even with such a low prevalence. Additionally, although SQLi has the lowest prevalence rate, its presence in some of the web platforms covered in this case study is proof of the fact that the patch-up approach that is often taken to fixing vulnerabilities that may be perceived at the network layer fall short of effectively mitigating these vulnerabilities, thereby allowing them to propagate throughout infrastructure. In-between the highest and lowest prevalence rates are XSS and BASM. The XSS flaw is the second highest in occurrence and the third most critical whereas the BASM is the third most occurring but the second most critical, according to the work in [4]. The vulnerabilities have associated occurrences of 523 and 378 respectively across all platforms. Some practical BASM flaws encountered at the web application layer through static code analysis are as follows.

- Session Cookie without secure flag set.
- Session Cookie without HTTP flag set.
- Rails Cookie persistence.
- User credential storage in cleat text.

While some of these application level security risks can be attributed to programming skill, sound security routines at various levels of the SDLC are still sufficient in detecting and mitigating such security loop holes and safeguard critical user data and privacy.

Drawing from the overall data obtained in this case study as well the correlation presented in Figure 5, it can be concluded that web application vulnerabilities remain a major threat to organizations regardless of their size or business model. Furthermore, because most of the highly severe vulnerabilities could be attributed to programming skill and practice, the pressure placed upon development teams could be a factor contributing to the high prevalence rate of web vulnerabilities.

In conclusion, although the web interface is only one of the exploitable components of an infrastructure, due to the nature in which this interface is openly placed within the public domain, it becomes therefore very unchallenging for attacks on it to become rapidly spread into some other components of the organization's infrastructure. While this is a well known fact within the web development and security community, the results presented here in the case study suggest that little is being done to protect this sensitive web interface. This phenomenon could be attributed to the fact that web development teams are often excessively focused on the realization and implementation of algorithms and engineering solutions for the web, with very tight deadlines, leading to an oversight in the security aspect of these

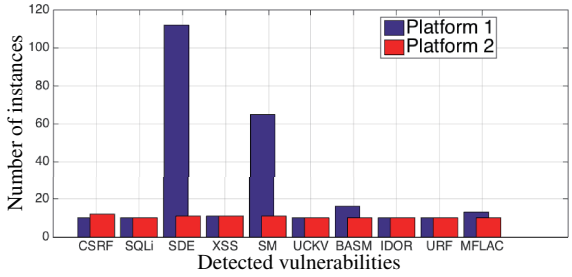

Figure 5 An illustration of the most vulnerable (blue) and least vulnerable (red) platforms. All values are off-set with 10 points for clarity of illustration.

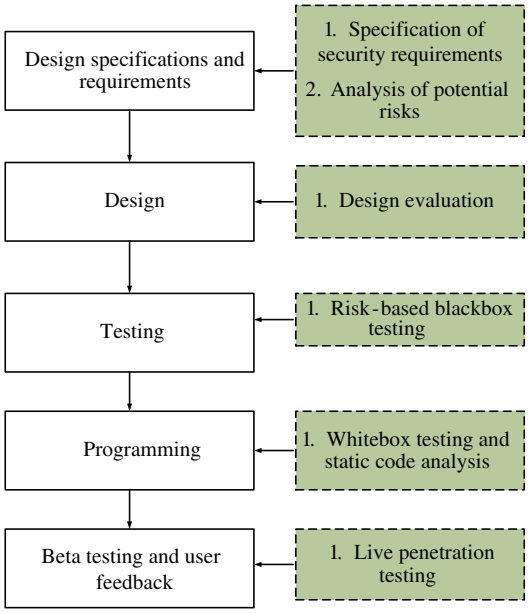

Figure 6 A straight-forward approach to security-oriented SDLC. Green components represent proposed additional operations.

web applications. That is to say that more emphasis is being placed on design rather than on security. The role of automated security audit, such as the ones carried out here in this case study could therefore not be exaggerated. While these automated audits do not offer a one-stop solution to web security, they provide a cost-effective and scalable approach to security awareness. This awareness, coupled with sound routines at each layer of the SDLC, provides a robust strategy in ensuring web application security.

Based on the experimental results obtained through the diversified case study conducted within Saudi Arabia, some recommendations and mitigation schemes are proposed towards safe and reliable security-conscious software development for web and cloud services.

5.2 Safeguarding cloud software security and integrity

The SDLC is a mechanism sufficient in guaranteeing the security of software developed at the application layer for both web and cloud services. The paper argues that by ensuring security awareness at each stage within the SDLC, cloud service providers become more efficient in safeguarding user data and service integrity within their online frameworks. Figure 6 presents a modified SDLC structure with proposed additional components introduced at each stage in the pipeline towards ensuring overall service quality and integrity that propagates all the way through the design stage and continues to hold even for deployed online systems.

The following mechanisms are proposed towards ensuring security and integrity within web and cloud service frameworks.

5.2.1 *Security awareness*

The first and most crucial step in safeguarding security at each step of the SDLC is the through a well engrained security awareness within the organizational operation. With security at the fore-mind of each staff within the development team, security-oriented development of software becomes the order of business in an environment where security becomes the benchmark of all operations. With this condition satisfied, it can be expected that all staff within a development will act consciously to safeguard web service security and in situations where the skills required in ensuring security are absent, security consultants need to be engaged to supplement the efforts of the already security aware development team.

5.2.2 *Application-centered development*

With the growing complexity of web and cloud services, there has been an accompanying growth in the various technologies and scripts that are relied upon in the realization of such complex systems. For this reason, in the development of new systems or in the extension of already existent ones, a thorough understanding of both underlying technology as well as new potential technology needs to be attained. Priority needs to be placed on the application development goals and purposes in order to allow for efficient and workable security oriented SDLC routines to be realized at each level. The integration of new components into already complex systems needs to be performed in a manner that does not further introduce security flaws into current infrastructure.

5.2.3 *Reliable threat modeling*

While threat modeling has been a well explored security mitigation scheme in recent years, the need for reliable and well representative threat models is still an issue that merits further effort. Reliable models need to provide a perspective of which security threats and loopholes could be encountered in both pre-deployment and post-deployment stages of web software development. In attaining reliable models for web and cloud software design, the need for effective categorization schemes such as the OWASP scheme adopted in the work in this paper are crucial. Such categorization schemes help in modeling threats in an efficient manner, thereby aiding in the forecast of potential security threats that may surface in an accidental or motivated manner. Accidental threats may include the accidental discovery of bugs and sensitive software information while motivated threats may include script kiddies or organized crime syndicates.

Conscious and well established threat modeling has the benefit of becoming more robust over time as it builds upon well established and tested threat models with newly emerging ones. In the area of threat modeling, the OWASP [4] model remains one of the most efficient and highly effective points of reference.

5.2.4 *Post deployment security auditing*

In modern day systems, the launch of a platform does not constitute the end of the development life cycle but a new stage in the evolution of the software since online updates and modifications of the already launched system are common ways of extending platform features without suffering service downtime. For this reason, post-deployment security audit has become one of the most common means by which security is ensured within the enterprise ecosystem.

In contrast to the design stage where data flows and code analysis are heavily relied upon, the most effective means of ensuring post-deployment security remains penetration testing [21]. Penetration testing could either be realized through the use of automated tools or via static code analysis, in line with the work presented here in the paper. While static code analysis relies heavily on the skill of development of security audit personnel, automated penetration testing is easily realized by means of tools and scripts including but not limited to those that are applied in the realization of the experimental case study presented here in this paper. Acunetix [18], Netsparker [20] and Nikto [19] are very good examples of automated tools that are applicable in realizing sound and effective penetration testing in the post-deployment stages of

web and cloud software development. The choice of tools however bears a strong correlation with the target platform properties as well as monetary resources available towards the realization of the task.

While automated tools offer a rapid, cost effect and reliable means of detecting vulnerabilities within complex platforms in a black-box approach, the deficiency associated with these tools remains an ongoing research effort and one that merits further research. While a thorough modeling of the draw-backs associated with these tools falls outside the scope of this paper, we address some general modeling of the drawback problem. Firstly, the efficiency of these automated tools is linearly correlated with the stage of platform development. This correlation is established by the fact that, while automated tools may be used to establish the security state of a platform in its early stages of development, future platform upgrades and patches may open up the platform to vulnerabilities in new ways that were otherwise not present in early stages. In relying on automated tools, it is therefore necessary to perform security audits at each stage of development. Secondly, automated tools, while more efficient than manual analysis, still require a thorough capability and know-how regarding cloud and web security. For this reason, automated security audits may completely fail to secure a platform if they are simply executed in a plug-and-play manner without a thorough understanding of the target platform and the various vulnerabilities that it may likely be suffering from. Finally, since automated tools are software development projects themselves, they need to be kept up to date in order to maintain their operational efficiencies. As zero-day vulnerabilities are detected, an automated tool will require a timely update in order to acquire vulnerability signatures or will otherwise simply fail. Unfortunately, within current trends, trade-offs are made at various stages within the SDLC that impair the efficiencies of these tools which are immensely relied upon in platform security auditing.

Regardless what tools are adopted, static code analysis in white-box environments is strongly recommended towards a cross-verification and supplementation of the security state of online systems. An approach that strongly compensates for draw-backs in automated tools.

6 Conclusion

Critical web and cloud application quality is inseparable from application security. Security remains a core component of cloud services as more services continue to migrate to such environments. The growing scale and complexities of web platforms has been accompanied by an equally growing and unrelenting effort to breach and compromise user data that is entrusted in the hands of these online platforms. While some of these breaches could be attributed to other factors including insider data and access leaks, the lack of quality and efficient security-oriented standards along the SDLC is to blame for a majority of these security compromises in modern day ecosystems. Towards addressing the issue, the work in this paper conducts an experimental case study targeting 70 web services in Saudi Arabia. Automated tools and static code analysis are applied towards an online evaluation of the security conditions of these platforms in direct and indirect test bed approaches. The results obtained through these highly diversified online experiments suggest that, while some security mechanisms can be achieved through programming skills and models, the greater aspect of application layer software security flaws are strongly linked with the lack of stringent and efficient security standards throughout the SDLC of these services.

Drawing from the experimental results, the paper proposes straight-forward and effective mechanisms through which application security can be safeguarded through-out the entire development life cycle. A core contribution of this paper lies in its thoroughness in applying real-time experimental data in extrapolating the true security conditions of a large number of platforms. This highly diversified experimental approach to web and cloud security analysis facilitates the efficient proposal of highly effective and easily applicable mitigation and preventive schemes. Unlike previous work that propose highly theoretical and unpractical solutions, the proposals put forward in this paper are highly practical and in line with modern day cloud development trends. While this paper mainly focuses on vulnerabilities and tools are adopted as a means of detection and mitigation, future work will focus on the automated security tools themselves in a way that deeply analyzes the tool behavior. This back-end analytical information on the

tools will be beneficial to the security and research community by providing a deep understanding on why certain security vulnerabilities may be missed in early stages of platform development, as well as what the leading factors that drive false-positive tool reports may be.

Acknowledgements This work was supported by Ministry of Higher Education in Saudi Arabia and National Basic Research Program of China (Grant No. 2014CB340600). Many thanks to the team from Cluster and Grid Computing Lab at Huazhong University and the staff from the Saudi Culture Mission in China for their immense support towards this research work.

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Awoleye O, Blessing V, Ilori A. Web application vulnerability assessment and policy direction towards a secure smart government. Government Inf Quarterly, 2014, 31: 118–125
2 Garber L. Security, privacy, and policy roundup. IEEE Secur Priv, 2012, 10: 15–17
3 James T, Khansa L, Cook D, et al. Using network-based text analysis to analyze trends in Microsoft's security innovations. Comput Secur, 2013, 36: 49–67
4 Razzaq A, Anwar Z, Ahmad F, et al. Ontology for attack detection: an intelligent approach to web application security. Comput Secur, 2014, 45: 124–146
5 Zhu Z, Zulkernine M. A model-based aspect-oriented framework for building intrusion-aware software systems. Inf Softw Tech, 2009, 51: 865–875
6 Armbrust M, Fox A, Griffith R, et al. A view of cloud computing. ACM Commun, 2010, 53: 50–58
7 Ludinard R, Totel E, Tronel F, et al. Detecting attacks against data in web applications. In: Proceedings of the 7th International Conference on Risk and Security of Internet ans System, Cork, 2012. 1–8
8 Zhang H G, Han W B, Lai X J, et al. Survey on cyberspace security. Sci China Inf Sci, 2015, 58: 110101
9 Ramachandran M, Chang V. Recommendations and best practices for cloud enterprise security. In: Proceedings of IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), Singapore, 2014. 983–988
10 Chess B, McGraw G. Static analysis for security. IEEE Secur Priv, 2004, 2: 76–79
11 Zhuan Y, Gessiou E, Portzer S, et al. Netcheck: network diagnoses from blackbox traces. In: Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), Seattle, 2014. 115–128
12 Dukes L, Yuan X, Akowuah F. A case study on web application security testing with tools and manual testing. In: Proceedings of IEEE Southeastcon, Jacksonville, 2013. 1–6
13 Mei J J. An approach for sql injection vulnerability detection. In: Proceedings of the 6th International Conference on Information Technology: New Generations, Las Vegas, 2009. 1411–1414
14 Patel S, Rathod V, Prajapati J. Comparative analysis of web security in open source content management system. In: Proceedings of International Conference on Intelligent System and Signal Processing, Gujarat, 2013. 344–349
15 Zhang Y, Liu Q, Luo Q, et al. XAS: cross-API scripting attacks in social ecosystems. Sci China Inf Sci, 2014, 58: 012101
16 Hashizume K, Rosado D, Fernández E, et al. An analysis of security issues for cloud computing. J Int Serv Appl, 2013, 4: 1–13
17 Behl A. Emerging security challenges in cloud computing: an insight to cloud security challenges and their mitigation. In: Proceedings of World Congress on Information and Communication Technologies (WICT), Mumbai, 2011. 217–222
18 Muscat I. Web vulnerabilities: identifying patterns and remedies. Netw Secur, 2016, 2016: 5–10
19 Davies P, Tryfonas T. A lightweight web-based vulnerability scanner for small-scale computer network security assessment. J Netw Comput Appl, 2009, 32: 78–95
20 Saleh A, Rozali N, Buja A, et al. A method for web application vulnerabilities detection by using boyer-moore string matching algorithm. Procedia Comput Sci, 2015, 72: 112–121
21 Antunes N, Vieira M. Penetration testing for web services. Computer, 2014, 47: 30–36