Received November 6, 2019, accepted November 19, 2019, date of publication November 29, 2019, date of current version December 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2956770

Systematic Literature Review on Penetration Testing for Mobile Cloud Computing Applications

AHMAD SALAH AL-AHMAD[1], HASAN KAHTAN[2], FADHL HUJAINAH[2], AND HAMID A. JALAB[3]

[1]Management Information Systems Department, College of Business Administration, American University of the Middle East, Egaila 15453, Kuwait
[2]Faculty of Computing, University Malaysia Pahang, Kuantan 26300, Malaysia
[3]Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur 50603, Malaysia

Corresponding author: Ahmad Salah Al-Ahmad (ahmad.alahmad@aum.edu.kw)

This work was supported in part by the Research Fund of American University of the Middle East in Kuwait, and in part by the Department of Research and Innovation, University Malaysia Pahang, under Grant RDU190189.

ABSTRACT Mobile cloud computing (MCC) enables mobile devices to exploit seamless cloud services via offloading, and has numerous advantages and increased security and complexity. Penetration testing of mobile applications has become more complex and expensive due to several parameters, such as the platform, device heterogeneity, context event types, and offloading. Numerous studies have been published in the MCC domain, whereas few studies have addressed the common issues and challenges of MCC testing. However, current studies do not address MCC and penetration testing. Therefore, revisiting MCC and penetration testing domains is essential to overcoming the inherent complexity and reducing costs. Motivated by the importance of revisiting these domains, this paper pursues two objectives: to provide a comprehensive systematic literature review (SLR) of the MCC, security and penetration testing domains and to establish the requirements for penetration testing of MCC applications. This paper has systematically reviewed previous penetration testing models and techniques based on the requirements in Kitchenham's SLR guidelines. The SLR outcome has indicated the following deficiencies: the offloading parameter is disregarded; studies that address mobile, cloud, and web vulnerabilities are lacking; and a MCC application penetration testing model has not been addressed by current studies. In particular, offloading and mobile state management are two new and vital requirements that have not been addressed to reveal hidden security vulnerabilities, facilitate mutual trust, and enable developers to build more secure MCC applications. Beneficial review results that can contribute to future research are presented.

INDEX TERMS Mobile cloud computing, penetration testing, offloading, mobile testing, cloud testing.

I. INTRODUCTION

Mobile cloud computing (MCC) is defined as a computing model that consists of mobile and cloud computing services via the Internet [1]. MCC represents the integration and convergence of these two technologies into a single seamless model. Although this integration has immense advantages, it has also exacerbated security and complexity, including those of MCC applications [2]. These applications run portable devices and harness the power and availability of cloud services to complete tasks such as accelerated cloud processing power and unlimited storage [3]. MCC applications use cloud services by offloading mobile application tasks to the cloud based on a set of conditions and criteria related to the respective mobile device states, running tasks and cloud status [4].

Thus, MCC applications are unique and complex, and testing them is difficult because of the various execution paths and locations of each process, which represent different offloading implementations [5]. Each test case for a MCC application needs to be generated and executed with respect to the device state to cover all possible process flows [6]–[8]. Therefore, the number of generated test cases is substantially increased [9].

MCC applications inherit the issues and characteristics of both cloud computing and mobile computing, [10], [11], which forces the testers to test cloud computing, mobile

The associate editor coordinating the review of this manuscript and approving it for publication was Eyuphan Bulut.

computing, large data, mobile device camera and sensors, heterogeneity, critical usage and offloading [12]–[14]. For these reasons, testers need to test for both mobile and cloud vulnerabilities, i.e., mobile network performance and context events types.

Penetration testing of MCC applications can help to safely remove threats, protect resources, data and applications, improve user trust in MCC applications, reduce cost, and provide security level proof [15]–[20]. The testing involves certain phases, which vary depending on the application that is being tested and the testing model employed by the tester. Penetration testing is considered to be a mechanism that provides proactive security protection [21] by highlighting security issues [22].

Penetration testing is prevalent in all software security practices [23], [24] and was defined as the art of finding an open door in 2002 [25]. Penetration testing is a postdeployment vulnerability assessment task that is conducted as an isolated test process in a manual and even ad hoc manner [26]–[29]. Penetration testing is used as a testing methodology to overcome security issues and fit with requirements for mobile, cloud, and web applications [18], [30].

In general, penetration testing examines systems using multiple attacks and attempts to find and exploit vulnerabilities using appropriate malicious input values that help to discover security bugs in implementation [31], [32]. The majority of penetration testing models included the phases of planning, analysis, test case generation, test case selection, test case execution, and reporting [33].

Penetration testing is an important task that includes many phases. Penetration testing is essential for all types of applications, including MCC applications. According to the complexity and uniqueness of MCC applications, the phases of MCC applications penetration test from planning to reporting are affected and become more complex [7], [8], [34], [35].

Although a few works have attempted to address MCC testing few studies have researched MCC application penetration testing. Similarly, our review of penetration testing models revealed that these models disregarded offloading and none of them addressed mobile, cloud, and web vulnerabilities. Therefore, requirements for a penetration testing model for MCC applications are needed.

To investigate the strengths and limitations of MCC testing, few review studies have been conducted, e.g., [5], [36], [37]. The main concern of related studies was to provide an overview of software or MCC testing, with the exception of the studies by Akherfi *et al.* [4], which focused on penetration testing. Although they are useful, existing studies lack sufficient consideration for penetration testing within MCC applications. This research is one of the first studies to perform a systematic literature review (SLR) on the research field of penetration testing in MCC apps.

Consistent with previous studies, our SLR is concerned with exploring the significant impact of penetration testing of MCC applications and analyzing the requirements required for testing MCC applications. This SLR study focuses on identifying current penetration testing models for mobile, cloud and MCC by presenting a critical investigation of each model in terms of the process description, advantages and constraints. A thorough analysis of the listed penetration testing models is executed to analyze their abilities in handling the complexity and uniqueness of MCC. A discussion of the challenges of penetration testing within MCC applications and future trends are discussed to assist practitioners and researchers in solving the defined challenges.

This paper is structured as follows. Section 2 discusses the research methodology in the paper. Section 3 demonstrates the testing requirements of MCC applications. Section 4 presents a discussion of the related work on penetration testing for mobile, cloud, and MCC. Section 5 presents the findings of the review. Section 6 concludes the paper.

II. MOTIVATION AND RELATED WORK

Various factors encouraged the authors to perform this SLR. No other research review has focused on the topic of penetration testing of MCC applications. Although various review studies have addressed penetration testing and MCC, none have focused on the areas of penetration testing of MCC applications. Penetration testing of MCC applications is a forthcoming research trend in penetration testing and MCC domains. This trend motivated this SLR research.

A literature review in any form enables the identification of gaps and challenges that exist in specific research themes, such as penetration testing of MMC applications, thus providing a comprehensive overview of the entire state-of-the-art in this domain for researchers and practitioners. TABLE 1 presents a summary of related existing review studies with respect to their focus and findings in comparison to this SLR research.

Ahmad *et al.* [36] conducted a systematic mapping study to review the software cloud testing methods from 2010-2015. The review reported on existing nonfunctional and functional cloud testing methods. In addition, the application and usage of these identified methods were precisely analyzed and discussed.

Anitha and Srinath [38] presented a review of the software testing models in the area of cloud computing. Empirical evidence from existing software testing techniques in cloud computing and a specific analysis of these models with respect to their performance and quality measurements are provided in this review. Similarly, another review study of software testing in the cloud computing platform was documented in [37]. This review provided an explanation of software testing in cloud computing and a discussion of existing cloud computing testing techniques.

Complementing the work of Al-Ahmad *et al.* [5], who studied mobile testing and MCC testing and provided a description of mobile testing with respect to its features and existing models. MCC testing was discussed in terms of mobile cloud testing issues, and a description was presented of the importance of the testing process in MCC applications. Recently, Al Shebli and Beheshti [39] conducted a review

TABLE 1. Summary of related studies.

Study Reference	Study Focus/Findings	Findings Similar to the SLR Results	Findings Added to this SLR
Ahmad et al. [36]	Software cloud-based testing with respect to an examination of available cloud-based testing models	Overview of the testing process in cloud computing.	Overview of the importance of executing the penetration testing process for a MCC application. Detailed investigation of the requirements needed for testing a MCC application. Empirical evidence of existing penetration testing models. Detailed analysis of the process description, benefits, and limitations of each penetration testing model. Specific analysis regarding the ability of the identified penetration testing models to address the uniqueness and complexity of MCC applications.
Al-Ahmad et al. [5]	Mobile testing, MCC testing	Overview of mobile cloud computing testing. Overview of the significance of the testing process in a mobile cloud computing application.	Overview of the importance of executing the penetration testing process for a MCC application. Detailed investigation of the requirements needed for testing MCC applications. Empirical evidence of existing penetration testing models. Detailed analysis of the process description, benefits, and limitations of each penetration testing model. Specific analysis of the ability of the identified penetration testing models to address the uniqueness and complexity of MCC applications.
Anitha et al. [38]	Empirical evidence of the software testing techniques in cloud computing	Description of software testing in cloud computing applications.	Overview of the importance of executing the penetration testing process for MCC applications. Detailed investigation of the requirements needed for testing MCC applications. Empirical evidence of existing penetration testing models. Detailed analysis of the process description, benefits, and limitations of each penetration testing model. Specific analysis regarding the ability of the identified penetration testing models to address the uniqueness and complexity of MCC applications.
Al Shebli et al. [39]	Penetration testing with respect to the process, available software and tools used in performing the penetration test and application roles of the penetration test in the firm	Description of the penetration testing process. Importance of penetration testing.	Overview of the importance of executing the penetration testing process for a MCC application. Detailed investigation of the requirements needed for testing a MCC application. Empirical evidence of existing penetration testing models. Detailed analysis of the process description, benefits, and limitations of each penetration testing model. Specific analysis regarding the ability of the identified penetration testing models to address the uniqueness and complexity of MCC applications.
Siddiqui et al. [37]	Software testing in cloud computing platforms	Overview of software testing in cloud computing.	Overview of the importance of executing the penetration testing process for MCC applications. Detailed investigation of the requirements needed for testing the MCC application. Empirical evidence of existing penetration testing models. Detailed analysis of the process description, benefits, and limitations of each penetration testing model. Specific analysis regarding the ability of the identified penetration testing models to address the uniqueness and complexity of MCC applications.

of penetration testing and presented a detailed analysis of penetration testing in term of its process, available software and tools for conducting a penetration test, and application roles in the organization.

As observed in TABLE 1, most related studies were focused on providing an overview of software or MCC testing with the exception of Al Shebli and Beheshti [39], who focused on penetration testing. Although useful, these existing studies do not sufficiently consider penetration testing of MCC applications. Thus, this work is the first study to perform a SLR of the specific research domain of penetration testing of MCC applications.

Complementing existing work, our SLR aims to investigate the significant influence of executing penetration testing on a MCC application and analyzing the requirements for testing MCC applications. Additionally, this SLR research focuses on identifying existing penetration testing models and analyzing each identified model in terms of its process description, benefits and limitations. A detailed analysis regarding the ability of the identified penetration testing models to address the uniqueness and complexity of MCC applications is conducted and reported in this SLR. Furthermore, a discussion of the total challenges of penetration testing within MCC applications and an elaboration of future sets for supporting researchers and practitioners in addressing the identified challenges.

III. RESEARCH METHODOLOGY

In this research, the SLR methodology is selected to guide the review process [40]. Figure 1 presents the review protocol of this work, which was developed based on the standard SLR guidelines of Kitchenham and Charters [40]. The review protocol consists of three main phases: planning, conducting, and reporting. Each phase contains a certain number of stages. The planning phase consists of identifying the need to conduct this review and formulate research questions. In the second phase, a review is conducted, and it has a certain number of stages: search process, study selection procedure and data extraction and synthesis. The reporting phase involves a discussion of the results. A discussion of the implementation of each phase and its associated substages is distinctly elaborated in the following subsections.

A. PLANNING

The motivation for conducting this review was discussed in the previous section (motivation and related work), and previous review studies were analyzed in terms of their focus and findings. The analysis showed that recent SLRs have not investigated the area of penetration testing of MCC applications. This SLR emphasizes this gap by identifying and analyzing the significance of penetration testing of MCC applications, requirements for testing MCC applications and existing penetration tests in terms of their limitations, benefits, process description, and ability to address the uniqueness and complexity of MCC applications. The research questions of this SLR were formulated based on the defined objective. TABLE 2 presents the formulated research questions and the motivation associated with each research question.

B. CONDUCTING

This phase is executed with the following defined stages: search process, study selection, and data collection and synthesis. The implementation of these stages is explained in the following subsections.

1) SEARCH PROCESS STRATEGY

A well-defined search process has a key role in acquiring satisfactory quality and reliable results [6]. In this research, the search process is carefully performed to extract and collect all related existing studies to the specified domain based on two elements: search strings and resources. The search terms are formulated in this SLR based on the listed research questions and standard procedure in [40], [41], which consists of the following steps.

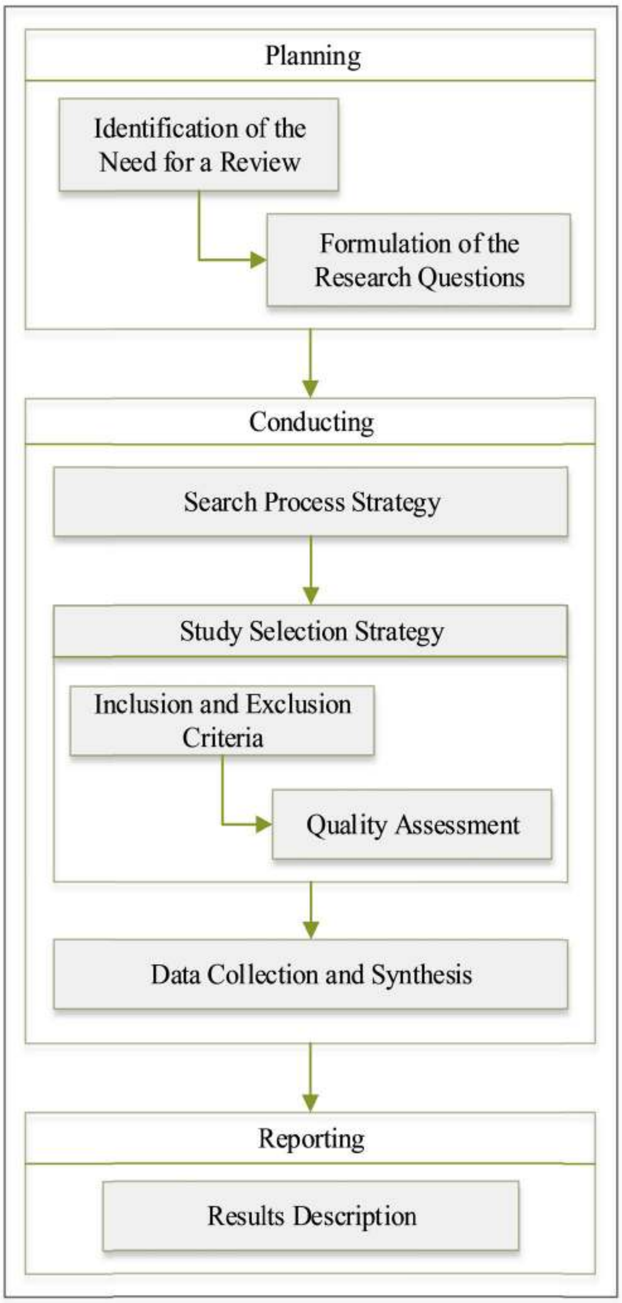
Planning
Identification of the Need for a Review
Formulation of the Research Questions
Conducting
Search Process Strategy
Study Selection Strategy
Inclusion and Exclusion Criteria
Quality Assessment
Data Collection and Synthesis
Reporting
Results Description

FIGURE 1. Review protocol.

1) Identifying the key terms of these research questions.
2) Listing the key terms synonyms and spelling alternatives.
3) Validating the relevant study search terms.
4) Combining the search terms with Boolean OR/AND operators.

TABLE 2. Research questions.

Research Question (RQ)	Motivation
RQ1. What is the significance of performing penetration testing of MCC applications?	To investigate and identify the significant impacts of executing penetration testing of MCC applications, specifically emphasizing the extent to which the execution of penetration testing can produce high-quality MCC applications in real scenarios.
RQ2. What are the requirements for conducting penetration testing of MCC applications?	To extract the requirements that are needed to test MCC applications.
RQ3. What are the available models for penetration testing and how can we classify them?	To identify the existing techniques that have been proposed with a focus on performing penetration testing and propose a new category for the identified models that will be considered.
RQ4. What are the process description, benefits and limitations of each identified model and which aspects can be adopted for testing the MCC application?	To critically analyze each identified model with respect to their process description, benefits and limitations. The process description refers to the steps involved in executing penetration testing of MCC applications. Regarding the benefits, the salient properties of each identified model are discussed. Last, the limitations and challenges of each identified model are revealed and precisely elaborated.
RQ5. Do previous proposed penetration testing models and techniques address the uniqueness and complexity of MCC applications?	To examine the identified penetration testing models regarding the generated list of MCC applications testing requirements and determine whether the identified penetration testing models can be adopted to efficiently test the MCC applications to identify hidden vulnerabilities.
RQ6. What are the recommended improvements to address the revealed limitations?	To identify the recommended future sets to address the identified limitations.

The final list of the formulated search terms are presented as follows.

- Pen testing OR penetration testing.
- Mobile cloud computing AND (Pen testing OR penetration testing OR testing).
- (Mobile cloud computing application OR software) AND (pen testing OR penetration testing OR testing).
- Mobile cloud computing AND (pen testing OR penetration testing) techniques OR methods OR frameworks, OR approaches.
- Limitations (AND/OR) challenges (AND/OR) issues (AND/OR) benefits (AND/OR) advantages of pen testing OR penetration testing techniques OR methods OR frameworks, OR approaches.
- Uniqueness (AND/OR) complexity of mobile cloud computing (AND/OR) in penetration testing techniques OR methods OR frameworks, OR approaches.

In this SLR, the search process is conducted using certain electronic libraries. These libraries are well-known resources that include either empirical studies or literature surveys in the domains of software engineering [42]–[44]. The following electronic libraries are selected:

- Google Scholar
- ACM Digital Library
- IEEE Xplore
- ScienceDirect
- SpringerLink
- ISI Web of Science
- Wiley InterScience
- Inspec
- Scopus

2) STUDY SELECTION STRATEGY

The study selection strategy has to be defined at the protocol to select the most relevant studies in the defined scope [6]. By conducting a defined search process, 500 papers were collected. To filter these collected studies to identify the most relevant studies in the defined domain of this research, the selection process is conducted within two stages: inclusion and exclusion criteria application and quality assessment.

The inclusion and exclusion criteria are formulated in this research based on the listed research questions. TABLE 3 presents the defined inclusion and exclusion criteria. The collected research studies were filtered based on the defined rules of the inclusion and exclusion criteria. Studies that focus on the software penetration testing and cloud computing applications or cloud computing testing and studies that include at least one potential answer to the specified research questions based on analyzing their titles, keywords and abstracts were included.

Non-English research studies and studies concerning only hardware testing or networks and other fields (not related to the specified domain) were excluded. A duplicate analysis was conducted to remove duplicate studies that were retrieved, and a recent copy was included. Thus, 117 studies were selected as the filtering results of the execution of the inclusion and exclusion criteria.

Concerning the quality assessment stage, quality checklist questions were formulated based on the defined research questions and guidelines of Chandane and Bartere [6]. The title, abstract and full content of each study of the 117 studies were precisely studied and evaluated according to the formulated quality checklist questions in TABLE 4. Each question

TABLE 3. Quality assessment inclusion and exclusion criteria.

Inclusion criteria	Exclusion criteria
Studies that are written in English.	Studies that are not written in English.
Studies that propose or include a model and architecture for penetration testing.	Studies that do not present sufficient technical details about the model or technique for penetration testing.
Studies that focus on software penetration testing and/or mobile cloud computing applications or cloud computing testing.	Studies that address hardware testing or networks and other fields that are not directly related to the specified domain.
Studies that report issues, problems, or any type of experience concerning software penetration testing and cloud computing applications or cloud computing testing.	Studies that do not concern software penetration testing and cloud computing applications or cloud computing testing.
Studies that are related to the specified research questions based on their title, keywords and abstract.	Gray studies (nonpeer-reviewed studies, work in progress, website studies, and studies that do not have bibliographic details).
	Studies that are not clearly related to any defined research questions.

TABLE 4. Quality assessment quality checklist questions.

No.	Question	Answer's score
QA1	Are the aims clearly stated?	Yes = 1, moderately = 0.5, No = 0
QA2	Is the study's context well demonstrated?	Yes = 1, moderately = 0.5, No = 0
QA3	Does the study focus on the specified domain of defined research questions?	Yes = 1, moderately = 0.5, No = 0
QA4	Is the proposed model/solution compressively explained?	Yes = 1, moderately= 0.5, No = 0
QA5	Is the evaluation of the proposed model performed on a competent benchmark data set or case study?	Yes = 1, moderately = 0.5, No = 0
QA6	Do the results add critical findings to the literature?	Yes = 1, moderately = 0.5, No = 0
QA7	Is the reporting clear and coherent?	Yes = 1, moderately = 0.5, No = 0

was scored as follows: yes = 1, moderately = 0.5, and No = 0; the total quality score of the study is the summation of the answers for all questions. These questions were answered individually for each study by each author, and meetings with all authors were conducted to calculate the total quality score for each study to reduce the probability of bias during the selection of related studies.

These studies, which have been selected by some authors but do not have a consensus, were collected to be reassessed and to answer the questions again by all authors in teamwork meetings to separately determine whether to include or exclude these studies. To ensure the dependability of the findings of this review, the study was not selected if it obtained a total quality score less than 5 (which is less than half of the total score of 12), and the study was selected if it obtained a total score greater than 5. As a result, 30 studies were selected as primary studies for this SLR. TABLE 5 presents the selected primary studies and their final total quality scores. TABLE 6 depicts the number of studies obtained during each distinct phase of our SLR from each repository.

3) DATA COLLECTION AND SYNTHESIS

In this SLR, the software EndNote is employed in the process of data collection and referencing. The data were extracted and collected based on the listed defined research questions, where each selected primary study was critically studied to obtain any data that can assist in addressing the questions. In the data synthesis step, the summarized proofs were collected from the data gathered from the selected primary study to answer the listed research questions.

IV. RESULTS AND DISCUSSION

A. RQ1: WHAT IS THE SIGNIFICANCE OF PERFORMING PENETRATION TESTING OF THE mcc APPLICATION?

Penetration testing is a postdeployment vulnerability assessment task that is conducted as an isolated test process in a manual and even ad hoc fashion [26]–[29]. This testing consists of finding an open door or bugs in the implementation of computer systems [25] that are installed to compromise the system security by attempting to attack the system and exploit its vulnerabilities [67]. The main characteristics of penetration testing are flexibility and scalability in testing for security vulnerabilities. Penetration testing is visible, can automate several of its processes and can operate independently of platforms and operating systems. Therefore, penetration testing is an ideal solution to the complexity, uniqueness, security, automation, and heterogeneous issues

TABLE 5. Quality scores for the results of primary studies.

Ref.	QA1	QA2	QA3	QA4	QA5	QA6	QA7	Score
[45]	1	1	1	0.5	1	1	1	6.5
[46]	1	1	1	1	0.5	1	1	6.5
[47]	1	1	1	1	1	0.5	1	6.5
[48]	1	1	1	1	0.5	1	1	6.5
[49]	1	1	1	1	1	0.5	0.5	6
[50]	1	1	1	1	0.5	1	1	6.5
[51]	1	1	1	1	0.5	0.5	1	6
[52]	1	1	1	1	1	0.5	1	6.5
[53]	1	1	0.5	1	1	1	1	6.5
[54]	1	1	1	1	1	0.5	0.5	6
[17]	1	1	1	1	1	1	1	7
[55]	1	1	1	1	0.5	1	1	6.5
[56]	1	1	1	1	1	1	1	7
[26]	1	1	1	1	1	1	0.5	6.5
[57]	1	1	1	1	0.5	1	1	6.5
[58]	1	1	1	1	1	1	1	7
[59]	1	1	1	1	0.5	1	1	6.5
[16]	1	1	1	1	1	1	1	7
[60]	1	1	0.5	1	1	1	1	6.5
[19]	1	1	1	1	1	1	0.5	6.5
[15]	1	1	1	1	0.5	1	1	6.5
[61]	1	1	1	1	1	1	1	7
[30]	1	1	1	1	1	1	1	7
[62]	1	1	1	1	0.5	1	1	6.5
[63]	1	1	1	1	1	1	0.5	6.5
[64]	1	1	0.5	1	1	1	1	6.5
[65]	1	1	1	1	1	1	0.5	6.5
[66]	1	1	1	1	1	1	1	7
[29]	1	1	1	1	0.5	0.5	1	6
[27]	1	1	1	1	0.5	1	1	6.5

TABLE 6. Number of studies obtained during all distinct phases of this SLR from all repositories.

Repository	Execution of the search process strategy phase	Execution of the inclusion and exclusion criteria phase		Execution of the quality assessment phase	
	Retrieved	Included	Excluded	Included	Excluded
Google Scholar	70	12	58	4	8
ACM Digital Library	171	22	149	8	14
IEEE Xplore	147	40	107	11	29
ScienceDirect	12	3	9	0	3
SpringerLink	42	8	34	3	5
ISI Web of Science	19	17	2	3	14
Wiley InterScience	8	3	5	0	3
Inspec	7	5	2	0	5
Scopus	24	7	17	1	6

observed when conducting security testing of any technology paradigm [17], [63].

MCC applications that consist of mobile and cloud components has increased the importance of conducting penetration testing because the number of users of mobile devices and mobile applications has substantially increased [68]. Therefore, enhancing the level of security for mobile devices and mobile applications by finding the hidden vulnerabilities using penetration testing is important [16], [58].

Furthermore, on-demand testing, scalable testing, and security testing are critical in cloud testing because the attacker can attack applications to gain access to a virtual machine (VM) or a cloud management system [18]. As previously discussed, penetration testing can be provided as a scalable [17], [26] on-demand service by external or internal penetration testers [36], [69], [70] and is one of the most common security tests types [23], [24]. Penetration testing of cloud computing is significant [30].

Disregarding penetration testing of MCC applications can omit security issues that may affect users, organizations, cloud providers and their customers. Due to a lack of penetration testing models and necessary tools [71] and the complexity of penetration testing [72], most organizations manually conduct penetration testing using external high-expertise testers [26]. This approach renders penetration testing expensive, time consuming, error prone and invisible in all SDL stages of application development [73].

B. RQ2: WHAT ARE THE REQUIREMENTS FOR TESTING MCC APPLICATIONS?

Testing MCC applications is a unique and distinct task [7], [8], [34], [35] since it uses offloading, which is complex and unique, and consists of multiple platforms and operating systems. The tester needs to test mobile and cloud vulnerabilities; mobile network performance will affect the test process as a factor; and several types of context events exist [7], [8], [34], [35]. These challenges are the five testing requirements that dominate and affect all phases of MCC applications penetration testing from planning to reporting as summarized in TABLE 7 [7], [8], [34], [35], [74]–[76].

Multiple platforms and operating systems, mobile network performance, and testing of multiple types of context events requirements as listed in TABLE 7 are specific for penetration testing or MCC. These requirements are general requirements for mobile, cloud, or web testing. However, the complexity of these requirements and issues increase when conducting penetration testing of MCC applications as previously discussed. Offloading and the need to test mobile, web and cloud vulnerabilities are specific to penetration testing of MCC applications.

These requirements have emerged because MCC applications are dynamically augmented tasks between mobile systems and the cloud using offloading. Thus, mobile and cloud computing use different platforms; for example, mobile systems can use an Android system and the cloud can use a Windows server. Therefore, testing these applications requires that the testers consider these offloading decisions and multiple platforms when generating, selecting and executing test cases.

The same rules apply for the types of vulnerabilities that need to be tested if we are evaluating mobile and cloud implementations. In addition to mobile and cloud vulnerabilities, we need to consider web vulnerabilities because they remain valid in MCC applications [7], [8], [34], [35]. Therefore, testing MCC applications must consider cloud, mobile, and web vulnerabilities while generating, selecting, and executing test cases because testers need to check for hidden vulnerabilities of all types.

The network in mobile events and other types of context events are additional variables that should be considered when testing MCC applications [7], [8], [34], [35]. Variations in the performance of this network may affect the offloading decision, which can change the flow and the executions path. Certain factors, such as receiving a call, receiving a SMS, shortages in memory, and overheated CPUs, and other actions that may occur while using the MCC application may change the execution path by affecting the offloading decision or firing default event handlers by the OS or application [7], [8], [34], [35]. Therefore, the surrounding environmental factors that may affect the network performance or interrupt the use of MCC applications need to be integrated when testing MCC applications to manage and control the applications.

C. RQ3: WHAT ARE MODELS ARE AVAILABLE FOR PENETRATION TESTING AND HOW CAN WE CLASSIFY THEM?

This formulated question consists of two facets: the first facet is related to the identification of existing techniques that have been proposed with the main focus of performing penetration testing of MCC applications. The second facets is the proposal of a new category for the identified models.

Multiple researchers build models to conduct penetration testing of web and service-oriented applications, such as [17], [19], [26]. These researchers have used penetration testing to test web applications, networks, desktop applications, web services, mobile applications and databases to identify hidden vulnerabilities that may be employed by attackers to harm applications, interrupt systems and steal data [19], [30], [49], [63], [81], [82].

Multiple penetration testing models have been investigated to determine the basic components of a model that can be used to conduct penetrating testing of MCC applications. This paper reviewed these models and classified them in terms of their components, target vulnerabilities, and features. The components used in the classification represents major module, phases, or steps of the investigated model. The target vulnerabilities are the type of vulnerabilities that this model can test. Similarly, features that represent the strengths of the model can be either identified in the literature or extracted from the results and logic. These models have been implemented and evaluated and provide a robust reference for penetration testing of other technologies. The models are listed in TABLE 8, and a detailed discussion is also provided.

As shown in TABLE 8 [17], [53], [63], [83], models are used to conduct penetration testing of networks as they target

TABLE 7. Testing requirements of MCC applications and the corresponding test phases that are affected.

Requirements	Description	Test phase
Offloading	Offloading in MCC affects the testing process due to multiple deployment models [74-76].	Test case generation, selection, execution, and reporting.
Multiple platforms and operating systems (Platform Independency)	Based on the specific mobile operation system, the model will choose which case to generate [7, 8, 34, 35].	Test case generation and reporting.
Automation	Penetration testing is a very expensive task in terms of cost and time. Therefore, automation is an important requirement [27, 62, 63].	Test case generation, selection, execution, and reporting.
Reporting	Well-organized and clear reporting will help to improve the process of fixing bugs and solving the identified vulnerabilities. Therefore, reposting is carefully investigated when proposing penetration testing model [29, 65, 66].	Reporting.
Use cloud services	Using cloud resources when conducting penetration testing of MCC applications will help to satisfy the extensive processing power requirements and provide scalability [7, 8, 34, 35].	Test case generation, selection, execution, and reporting.
Mobile state management	To execute any test case of MCC applications, the device state must be setup due to the offloading in MCC applications because it changes the execution path based on the mobile device state in some of its deployment models [7, 8, 34, 35]. Furthermore, a mobile agent needs to read the environmental variable from a mobile network to simulate the intended behavior of the real user [7, 8, 34, 35].	Test case execution and reporting.
The tester needs to test a. mobile vulnerabilities; b. web vulnerabilities; and c. cloud vulnerabilities	Each test case is generated to be injected in a mobile system or the cloud. Consequently, penetration testing of MCC applications should test mobile and cloud vulnerabilities [7, 8, 34, 35]. Web vulnerabilities also need to be considered because web attacks remain valid in the cloud-based application [20].	Test case generation, selection, execution, and reporting.
Different types of inputs and several types of context events	Several types of mobile device inputs, i.e., touch, text, GPS, camera, mic, and keyboard. Thus, penetration testing using these types of inputs is mandatory in the generated test cases. [7, 8, 34, 35]. Similarly, a test manager should record all parameters from different context event types [7, 8, 34, 35].	Test case generation and reporting.
Testing a SaaS	Penetration testing must be frequently conducted, which suggests that it can reduce the cost and improve the performance [77].	Test case generation, selection, execution, and reporting.
Scalability	Penetration testing of MCC applications must be scalable to handle large and complicated applications [17, 63].	Test case generation, selection, execution, and reporting.
Regression testing	Penetration testing needs to support consistent repeatable and continuous testing for frequently changing MCC applications [78].	Test case generation, selection, execution, and reporting.
Standardized	Standardization will help testers be consistent in conducting penetration testing of different MCC applications [63].	Test case generation, selection, execution, and reporting.
Use a graphical user interface (GUI)	Using a graphical user interface for the tool used in automating the penetration tests will improve the usability and input process and enhance the results presentation [19, 63].	Test case generation, selection, execution, and reporting.
Test for SOA/ APIs	MCC applications utilized offloading, which will delegate processes to the cloud by implementing SOA or using APIs. Therefore, penetration testing of MCC applications needs to address these types of application development methods [17].	Test case generation, selection, execution, and reporting.
Random test	Randomly generated input is a successful technique for penetration test case generation and test case selection [48, 52, 79, 80].	Test case generation and selection.

network vulnerabilities. While certain models target mobile applications, such as [49], [64], [84], because these models target the vulnerabilities of mobile applications. Similarly, some models have been designed to target web vulnerabilities, such as [15], [26], [28], [51], [61], [85], [86].

Similarly, [62] presents a model built to conduct penetration testing of cloud computing system vulnerabilities, including the platform and infrastructure, and [87] presents a model for conducting penetration testing of desktop applications and targets the most common desktop application vulnerabilities. These findings show that each technology is unique, which requires the construction of a specific model to conduct penetration testing.

These models are well defined, scalable, flexible, and clear, which render them usable and will improve their performance when employed by testers. The phases and steps for these models start with planning and end with reporting the identified vulnerabilities, although the phases do not always have the same name but are consistently applied. Within the model, all models provide a procedural description to generate the selected test cases and are executed to check for hidden vulnerabilities.

Among the previously listed models in TABLE 8, certain promising models can be extended to test MCC applications. The main nominated models are built to test web applications, i.e., [15], [26], [28], [51], [61], [85], [86]. Web penetration testing models were nominated because they target applications instead of services or networks, and web technology, including the back-end and front-end, can be mapped to mobile and cloud sites in MCC applications. Other application penetration testing models for desktops [87], cloud [62] and mobile devices [49], [64], [84] are designed to test applications parts that reside on either a desktop, cloud or mobile system. Furthermore, web models target web vulnerabilities, which remain valid in MCC applications [7], [8], [34], [35]. We discovered that the model proposed in [51] is the most promising model as it was well defined, scalable and successfully implemented in [15]. The model in [51] is well structured with clear steps that be extended to other technologies, especially in MCC applications.

D. RQ4: WHAT ARE THE PENETRATION TESTING PROCESS AND THE BENEFITS AND LIMITATIONS OF EACH IDENTIFIED MODEL, AND WHICH ASPECT CAN BE ADOPTED FOR TESTING A MCC APPLICATION?

The complexity and uniqueness of MCC applications in terms of multiple vulnerabilities domains and offloading must be addressed during test case generation, selection, and execution to address the maximum discovered application paths and identify mobile, cloud and web vulnerabilities with minimum time and cost by the penetration testers. By measuring the ratio of addressing MCC application penetration testing, improvements in the complexity and uniqueness will be represented by increasing the number of vulnerabilities and reducing the resources.

For instance, test preparation embodies an understanding of the application that is being tested [64]; therefore, it must incorporate the uniqueness and complexity of MCC applications. Similarly, penetration test case generation generates invalid, random, or unexpected inputs to a program to test for unseen vulnerabilities [92]–[94]. Thus, this approach can be considered a black box testing technique to identify flaws in software by feeding random input into applications and monitoring for crashes [92].

According to these definitions, multiple models and techniques are available in the literature and industry to conduct penetration test case generation, which is an effective means of increasing the quality and security of software and systems. The main objective of this process is to identify security-relevant weaknesses in the implementation that may cause the system under testing to crash or produce anomalous behavior [95]. The process seeks to identify vulnerabilities that can be exploited to break into or crash a system [96].

Although test case generation processes generate test cases that encompass all application paths and use all attributes and variable domains, ranges and types, running all generated test cases is not possible in most cases due to time and cost limitations [97]. Therefore, finding mechanisms that select test cases or prioritize the generated test cases is mandatory to reduce the number of resources that are required while maintaining high path coverage and fault detection ratios [98]. The test case selection process is a problematic issue, especially in penetration testing, due to the numerous paths, input types, input methods, environment factors, and vulnerability types. Researchers, industry experts, and practitioners have attempted to construct new mechanisms that enhance the ratio between the number of selected test cases versus the path coverage and fault detection ratios [99].

When generating testing cases, the direct implication of MCC application penetration testing introduces another testing variable, i.e., offloading, which affects the application execution paths because it exponentially increases the number of generated test cases [47], [100]–[104]. Therefore, the test case selection process, especially in MCC application penetration testing, is essential. This process will select specific test cases that reduce the required time and resources while retaining the accepted test cases coverage [77], [105].

The previously discussed requirements of the penetration testing of MCC are determined from testing requirements inherited from mobile and cloud platforms and other requirements unique to MCC applications [7], [8], [15], [17], [35], [48], [50], [52], [63], [77], [78]. The 17 requirements include Offloading, Platform Independency, Automation, Reporting, Mobile State Management, Cloud Service Use, Mobile Vulnerabilities, Web Vulnerabilities, Cloud Vulnerabilities, Different Input Types, SaaS Testing, Scalability, Regression Testing, Standardization, GUI Use, API Testing and Random Testing. TABLE 9 shows the 17 MCC application penetration testing requirements compared with the 30 testing models from 1998 to 2016.

E. RQ5: DO THE PREVIOUS PROPOSED PENETRATION TESTING MODELS AND TECHNIQUES ADDRESS THE UNIQUENESS AND COMPLEXITY OF MCC APPLICATIONS?

Based on the previous analysis of the penetration testing models, two main patterns are observed. First, these models disregard offloading and none of them simultaneously address mobile, cloud, and web vulnerabilities. Second, the model components can be categorized into four main phases for penetration testing, namely, preparation, test case generation, test case selection, and test case execution. These four phases are used to conduct penetration testing of MCC or other types of applications.

Penetration testing of MCC applications helps to efficiently identify critical security assurance issues and application vulnerabilities; promotes the safe removal of threats; protects resources, data and applications; improves user trust

TABLE 8. Penetration testing models/architectures analysis.

Ref	Components	Vulnerabilities	Features
[63]	Network Entity XML, Exploit XML, GUI, Detect Module, Identify, Module, Predict Module, and React Module	Well-known network vulnerabilities	Flexibility Standardization Scalability
[17]	XML-based Platform, Multi-technology Resource Exploration Module, OVAL-based Vulnerability Assessment Module, Execution Engine, Results Analysis and Report, Generation Module, Shared and Vulnerability Database, and Penetration Testing Strategy Library	Network	Effective Performance Standard Diversity
[83]	Planning and Preparation Information Gathering and Analysis, Vulnerability Detection Penetration Attempt, Analysis and Reporting, and Cleaning Up	Network and software vulnerability shown by the scanner	Organized Scalable
[49]	Information, Team, Tools	Specific set of mobile vulnerabilities	Insights for tools and teams
[26]	Gray-box Test Architecture, Penetration Testing Process Integrated with Secure Software Development Life Cycle Web Application Penetration Test Security Model Test Campaign Process Automation.	Web vulnerability	Clear Detailed
[88]	Test Preparation Phase, Memorandum of Understanding, Non-Disclosure Agreement, Confidentiality Agreement, Vulnerability Assessment, Target Discovery, Scanning, Result Analysis, Reporting, Penetration Testing, Pre-Attack Phase, Attack Phase, and Post Attack Phase, Report Generation Phase	Web and network vulnerabilities	Phases are well defined
[85]	Data Gathering, Define Scope, Identify Common Vulnerabilities, Identify Tools, Scanning, Scanning Analysis, Risk Rating, Attacks, Test Case Development, Penetration Test, Reporting, Vulnerabilities Analysis Report	Web common vulnerabilities from OWASP	Specific steps for choosing tools and selecting common vulnerabilities
[89]	Flaw Hypothesis Model, Information Gathering, Flaw Hypothesis, Flow Testing, Flaw Generalization, Attack Tree Conjunctive and Disjunctive	Specific set of mobile vulnerabilities	Define the hypothesis before starting the test to reduce the time costs
[53]	Apply SVA Tools, Assessment Report Vulnerability Analysis, Risk Assessment, and Countermeasures Analysis	Network and servers	Includes selected tools to organize reports and fix vulnerabilities
[15]	Functional Test Cases Generator, Generated TTCN-3 Generator, Generated TTC-3 Functional Test Cases, Penetration Test Case Generator, Generated TTC-3 Penetration Test Cases, Generate TTC-3 Tests Execution Engine, and Test Reports	Web XSS and SQL injection	TTCN-3 engine and runtime environment that maximize reusability
[51]	Predefined Templates, Test Cases Attack Scenario, Test Case Engine, and TTCN-3 Runtime Environment	Web XSS and SQL injection	Well defined Detailed
[90]	Preparation, Anonymity, Foot-Printing Analysis, Exploitation, Advisor, and Reporting	Specific common set of vulnerabilities	Well-organized procedure
[84]	Test Management, Management Console, Web-Based Management GUI, Management App, Agents, Mobile Modem, and TCP/IP	Selected mobile vulnerabilities	User agents and remote management
[62]	National Vulnerability Database NVD, Ontology Knowledge Base OKB, System Classifiers, Indexer, Vulnerability Class Index, Semantic Natural Language Processor SNLP, Vulnerability List, and Attack Database	Cloud computing vulnerabilities	Well defined

TABLE 8. *(Continued.)* **Penetration testing models/architectures analysis.**

[61]	Building Model of the Penetration Test Case, Steal System Information, Error Message Utilizing, Blind Injection, Bypass Authentication, Remote Command Execution, SQL Command Injection, Non-SQL Command Injection, Instantiating the Penetration Test Case Model, Fingerprint of Web Application, And Certain Coverage Criteria	Web SQL injection	Applied security goal model (SGM), which identify additional vulnerabilities
[91]	Review Techniques, Documentation, Log, Rule Set, System Configuration, Network Sniffing, File Integrity Checking, Target Identification and Analysis Techniques, Network Discovery, Network Port and Service Identification, Vulnerability Scanning, Penetration Testing, Planning, Discovery, Attack, and Reporting	Specific set of network vulnerabilities	Adherence to the global security policies Confirm that all parties are covered
[87]	Planning and Preparation, Assessment Reporting, Clean-up and Destroy Artifacts	Desktop applications vulnerabilities	Use global security policies
[64]	Test Preparation Embodies an Understanding of the Application Under Test. Intelligence Gathering Demonstrates both the Environment and Architectural Analysis for Test Bed Applications in Terms of the Applications, Runtime Environment, Back-end Services, and Vulnerability Analysis	Mobile	Simplified the steps to conduct penetration testing of mobile applications
[86]	Identification of the Attacker Objectives, Definition of the Attacker Capability, Attacks Modeling, Attack Scenarios Generation, Attack Scenarios Implementation, and Transform Attack Scripts	Web services XSS	Well-defined steps to use tree model in case generation
[28]	Combine Multiple VAPT Tools (Open Source/Premium) and Apply Majority Voting and Weighted Priority	Web vulnerability	Cost-effective vulnerability analysis and penetration testing

with MCC applications; reduces costs; and obtains proof of the level of security [15]–[20]. Penetration testing can be conducted by testing models to run a systematic approach that has distinct phases, standards, and a quality control process.

Note that testing models are based on multiple testing methods. Of particular importance is the penetration testing method because it enables critical security assurance and vulnerability assessment testing for applications, networks, and web, mobile and cloud computing with minimum resources [16]–[20]. Furthermore, penetration testing is scalable and can be automatically executed and run on schedule without the need to stop the operational system [17], [26].

Due to the lack of models and necessary tools [71] and the complexity of penetration testing [72], most organizations manually conduct penetration testing using external high-expertise testers [26]. This approach renders penetration testing expensive, time consuming, error prone and invisible in all SDL stages of application development [73].

TABLE 10 illustrates the landscape of software penetration testing models in the domains of web, mobile, cloud and MCC as well as the general penetration testing models. This table consists of 30 application penetration testing models and frameworks between 1995 and 2016. The results show that multiple testing models and frameworks are used in general penetration testing, mobile penetration testing, or cloud penetration testing. However, none of these identified models are based on the domain of MCC application penetration testing.

All above penetration testing models disregard the uniqueness and complexity of MCC applications, which is required to identify MCC vulnerabilities. Thus, organizations that develop MCC applications either do not conduct penetration testing for these applications, use internal procedures that are not publicly published, or utilize expensive outsourced expert testers to test their applications [26], [106].

F. RQ6: WHAT IMPROVEMENTS ARE RECOMMENDED TO ADDRESS THE REVEALED LIMITATIONS?

The requirements in this paper were collected and analyzed to assess current testing models and evaluated as the basis for constructing a MCC application penetration testing model. Figure 2 shows the penetration testing requirements of MCC applications clustered with the respective sources of the aspect requirements, namely, MCC uniqueness, MCC complexity, MCC applications, general penetration testing, mobile computing penetration testing or cloud computing penetration testing. To mitigate the complexity and uniqueness of MCC application penetration testing, a testing model should be employed.

Testing models provide repeatability, automation, flexibility, platform dependency, regression test allowance, gray test support, easily readable results, cost efficiency, and full

TABLE 9. Current application testing models with requirements *1 = Offloading; 2 = Platform independency; 3 = Automation; 4 = Reporting; 5 = Mobile state management; 6 = Cloud service use; 7 = Mobile vulnerabilities; 8 = Web vulnerabilities; 9 = Cloud vulnerabilities; 10 = Different input types; 11 = SaaS testing in a cloud; 12 = Scalability; 13 = Regression testing; 14 = Standardization; 15 = GUI Use; 16 = SOA/API testing; 17 = Random testing R = References; TR = Testing requirements.

TR \ R	[45]	[46]	[47]	[48]	[49]	[50]	[51]	[52]	[53]	[54]	[17]	[55]	[56]	[26]	[57]	[58]	[59]	[16]	[60]	[19]	[15]	[61]	[30]	[62]	[63]	[64]	[65]	[66]	[29]	[27]
1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
2	×	×	×	×	×	✓	×	×	✓	×	×	✓	✓	×	✓	×	×	×	×	×	×	×	×	×	×	✓	×	✓	✓	×
3	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	×	×	✓	×	✓
4	✓	✓	×	×	×	×	✓	×	×	✓	✓	✓	✓	✓	×	✓	×	×	×	×	×	×	×	×	✓	×	✓	✓	✓	×
5	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
6	×	×	×	×	×	×	×	×	✓	×	×	×	✓	×	✓	×	×	×	✓	×	×	×	×	✓	×	×	×	✓	×	×
7	×	×	×	×	×	×	×	✓	×	×	×	×	×	×	×	✓	×	✓	×	✓	×	✓	×	×	×	✓	×	×	×	×
8	×	✓	✓	✓	✓	×	✓	×	×	×	✓	×	×	✓	×	×	×	×	×	×	✓	×	×	×	×	×	×	✓	×	✓
9	×	×	×	×	×	×	×	×	✓	×	×	×	×	×	×	×	×	×	✓	×	×	×	✓	✓	×	×	×	×	×	✓
10	×	×	×	×	×	✓	×	✓	×	×	×	×	×	×	×	×	✓	✓	×	×	✓	×	×	×	×	✓	×	×	✓	✓
11	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	✓	×	✓
12	✓	✓	✓	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	✓	✓	×	✓	✓	✓	✓	×	✓	×
13	✓	✓	✓	✓	×	✓	✓	×	×	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×	×	×
14	×	×	×	×	×	×	✓	×	×	×	×	×	✓	✓	×	×	×	×	×	×	×	×	×	×	✓	✓	×	×	✓	×
15	✓	✓	✓	✓	×	✓	✓	×	×	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×	✓	✓	×	✓	×	✓
16	×	×	×	×	×	✓	✓	×	×	×	✓	×	×	✓	×	×	×	×	✓	✓	×	×	×	×	×	✓	×	✓	×	×
17	×	×	×	✓	×	×	×	✓	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	✓	✓	×	×	✓

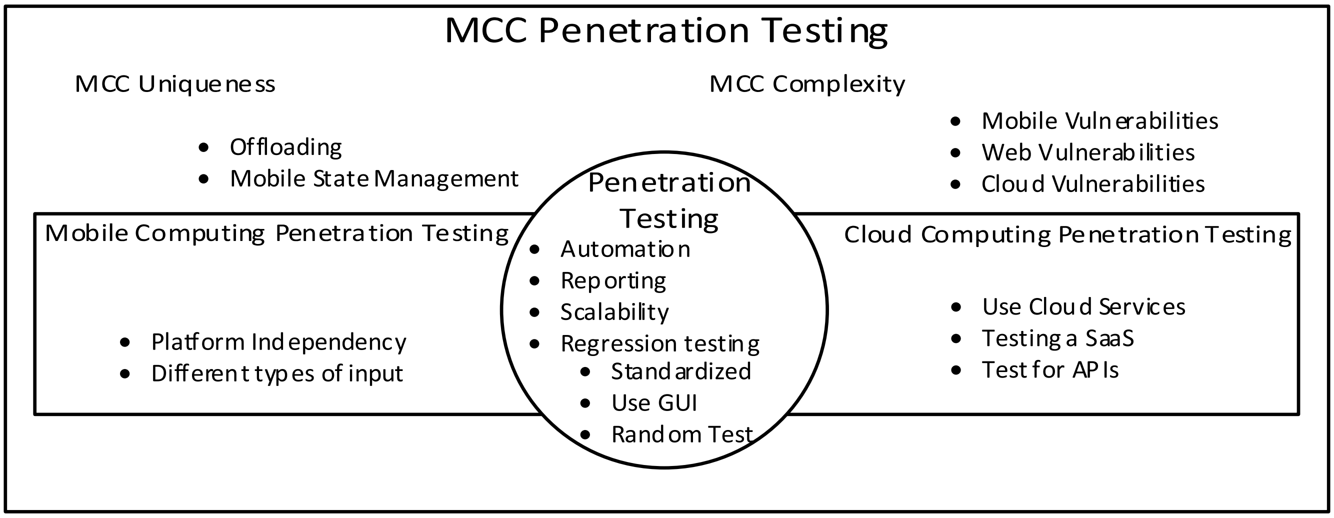

FIGURE 2. MCC application penetration testing source of the requirement.

integration with other system development life cycle (SDLC) processes (Xiong and Peyton, 2010). Consequently, the penetration testing model will reduce the complexity and uniqueness of penetration testing for MCC applications, increase the effectiveness when conducting penetration testing for MCC applications, and help testers conduct MCC application penetration testing.

For this reason, a model is needed that addresses offloading, multiple vulnerability domains, platform independence, automation, reporting, mobile state management, cloud service use, different input types, testing SaaS in a cloud, scalability, regression testing, standardization, GUI use, SOA/API testing, and random testing.

This model should generate, select, execute and report test cases that use offloading parameters, multiple types of inputs from mobile devices and networks; target APIs; apply random testing techniques; and apply general penetration testing standards. This model should provide scalable and

TABLE 10. **Domains of current penetration testing models *1 = General penetration testing; 2 = Web testing; 3 = Mobile testing; 4 = Cloud testing; 5 = Mobile cloud testing R=Reference; and T=Technology.**

T \ R	[45]	[46]	[47]	[48]	[49]	[50]	[51]	[52]	[53]	[54]	[17]	[55]	[56]	[26]	[57]	[58]	[59]	[16]	[60]	[19]	[15]	[61]	[30]	[62]	[63]	[64]	[65]	[66]	[29]	[27]
1	✓	✓	✓	✓	✓	×	✓	×	✓	×	✓	✓	×	×	×	✓	×	✓	×	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	×	×	×	×	×	×	✓	×	×	×	✓	×	×	✓	×	×	×	×	×	✓	✓	✓	×	×	×	×	×	✓	×	✓
3	×	×	×	×	×	✓	×	✓	×	×	×	×	×	×	✓	✓	✓	✓	×	×	×	×	×	×	×	✓	×	×	×	×
4	×	×	×	×	×	×	×	×	✓	✓	×	✓	✓	×	×	×	×	×	✓	×	×	×	✓	✓	×	×	×	×	×	✓
5	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

automated solutions that use cloud services with GUIs and can be delivered as a service to target mobile, cloud and web vulnerabilities in MCC applications.

This approach can be implemented by carefully examining previous penetration testing models to select and reuse the appropriate components. New processes that target the uniqueness of MCC applications are needed. These processes must follow the penetration standardization and be implemented in a set of tools that provides a scalable penetration testing service by using cloud capabilities. The penetration testing model should be generated by incorporating all the components used in test case generation, selections, and execution from previous models and then change them to meet the MCC penetration testing requirements. The existing testing models should be leveraged also to be used in test MCC applications by adding new components and modify old components to meet the MCC penetration testing requirements.

V. THREATS TO VALIDITY

Completeness, data synthesis, and publication bias represent the leading challenges of any systematic review [6]. To overcome the challenge of completeness, we employed a defined protocol that utilized a meticulous search strategy to systematically review penetration testing of MCC applications. In this protocol, 30 studies that were conducted from 2010 to 2018 were reviewed to answer the research questions. However, ensuring that every relevant study was reviewed is impossible. Additionally, studies that were published in a language other than English were not reviewed. Thus, some possibly relevant studies may have been overlooked. The challenge represented by data synthesis was satisfied by the quality assessment discussed in Section 3.1.2, which explained the process of synthesizing the data in this study. The quality assessment detected studies that contained information that could be used to answer the research questions. However, effective quality assessment was not guaranteed.

Publication bias occurs when positive results are emphasized to the detriment of any negative results [6]. To overcome any publication bias in this study, the literature that was reviewed was chosen using an extensive selection process. The selected studies were subjected to an in-depth quality assessment to confirm their relevance to the current study. Consequently, some studies that include technical reports, ongoing and unpublished studies, and non-peer reviewed studies (known as gray studies) were excluded even if they contained information that would answer at least one research question. The steps taken to prevent publication bias in this study also created one of its limitations because relevant information may have been excluded by omitting gray studies.

VI. CONCLUSION

MCC has exacerbated the security and complexity of MCC applications, and consequently, security penetration testing of these applications has become more complex and expensive. Penetration testers must identify possible hidden vulnerabilities that encompass mobile devices and network and cloud domains and include new parameters, such as platform and device heterogeneity, context event types and offloading. The addition of these new parameters to the test case generation exponentially increases the number of generated test cases.

This work critically reviewed the number of papers in the related areas of MCC, security and penetration testing. Previous studies have not addressed the problem of penetration testing for MCC applications, and none have proposed a MCC application penetration testing model. We have proposed a set of penetration testing requirements for MCC applications that can be used as the basis of such a model. In particular, offloading and mobile state management are two new and vital requirements that have not been addressed in penetration testing of previous technologies. The model envisages the discovery of hidden security vulnerabilities, facilitates mutual trust, provides security assurance, and enables developers to build more secure mobile cloud applications. This model should be built to target mobile, cloud and web vulnerabilities in MCC applications using standardized, scalable and automated solutions via a GUI and cloud services. Furthermore, the model must consider offloading parameters, multiple input types from mobile devices and networks, and target APIs and apply random testing techniques.

REFERENCES

[1] H. Qi and A. Gani, "Research on mobile cloud computing: Review, trend and perspectives," in *Proc. 2nd Int. Conf. Digit. Inf. Commun. Technol. Appl. (DICTAP)*, Bangkok, Thailand, May 2012, pp. 195–202.

[2] M. I. Sahu and U. Pandey, "Mobile cloud computing: Issues and challenges," in *Proc. Int. Conf. Adv. Comput., Commun. Control Netw. (ICACCCN)*, Oct. 2018, pp. 247–250.

[3] C. V. Raja, K. Chitra, and M. Jonafark, "A survey on mobile cloud computing," *Int. J. Sci. Res. Comput. Sci., Eng. Inf. Technol.*, to be published.

[4] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Appl. Comput. Informat.*, vol. 14, no. 1, pp. 1–16, 2018.

[5] A. S. Al-Ahmad, S. A. Aljunid, and A. S. A. Sani, "Mobile cloud computing testing review," in *Proc. Int. Conf. Adv. Comput. Sci. Appl. Technol. (ACSAT)*, Kuala Lumpur, Malaysia, Dec. 2013, pp. 176–180.

[6] S. H. Chandane and M. M. Bartere, "New computing paradigm: Software testing in cloud, issues, challenges and need of cloud testing in today's world," *Int. J. Emerg. Res. Manage. Technol.*, vol. 50, pp. 68–75, Feb. 2013.

[7] H. Muccini, A. Di Francesco, and P. Esposito, "Software testing of mobile applications: Challenges and future research directions," in *Proc. 7th Int. Workshop Autom. Softw. Test*, Zurich, Switzerland, Jun. 2012, pp. 29–35.

[8] B. Kirubakaran and V. Karthikeyani, "Mobile application testing—Challenges and solution approach through automation," in *Proc. Int. Conf. Pattern Recognit., Informat. Mobile Eng. (PRIME)*, Salem, India, Feb. 2013, pp. 79–84.

[9] A. S. Al-Ahmad and H. Kahtan, "Fuzz test case generation for penetration testing in mobile cloud computing applications," in *Proc. Int. Conf. Intell. Comput. Optim.*, 2018, pp. 267–276.

[10] C. Costea, "Applications and trends in mobile cloud computing," *Carpathian J. Electron. Comput. Eng.*, vol. 5, p. 57, Jan. 2012.

[11] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.

[12] B.-G. Chun, S. Ihm, P. Maniatis, and M. Naik, "Clonecloud: Boosting mobile device applications through cloud clone execution," 2010, *arXiv:1009.3088*. [Online]. Available: https://arxiv.org/abs/1009.3088

[13] S.-G. Kang, K.-W. Lee, and Y.-S. Kim, "Preliminary performance testing of Geo-spatial image parallel processing in the mobile cloud computing service," *Korean J. Remote Sens.*, vol. 28, no. 4, pp. 467–475, 2012.

[14] H. A. Turner, "Optimizing, testing, and securing mobile cloud computing systems for data aggregation and processing," Ph.D. dissertation, Virginia Tech, Blacksburg, VA, USA, 2015.

[15] B. Stepien, L. Peyton, and P. Xiong, "Using TTCN-3 as a modeling language for Web penetration testing," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Athens, Greece, Mar. 2012, pp. 674–681.

[16] T. Paananen, "Smartphone Cross-Platform Frameworks: A case study," M.S. thesis, Degree Programme Media Eng. School Technol., JAMK Univ. Appl. Sci., Jyväskylä, Finland, 2011.

[17] B. Xing, L. Gao, J. Zhang, and D. Sun, "Design and implementation of an XML-based penetration testing system," in *Proc. Int. Symp. Intell. Inf. Process. Trusted Comput. (IPTC)*, Huanggang, China, Oct. 2010, pp. 224–229.

[18] J. Gao, X. Bai, and W.-T. Tsai, "Cloud testing-issues, challenges, needs and practice," *Softw. Eng., Int. J.*, vol. 1, pp. 9–23, Sep. 2011.

[19] C. Mainka, J. Somorovsky, and J. Schwenk, "Penetration testing tool for Web services security," in *Proc. 8th IEEE World Congr. Servicess*, Honolulu, HI, USA, Sep. 2012, pp. 163–170.

[20] K. Karnad and S. Nagenthram, "Cloud security: Can the cloud be secured," in *Proc. 7th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, London, U.K., Dec. 2012, pp. 208–210.

[21] J. N. Goel and B. M. Mehtre, "Vulnerability assessment penetration testing as a Cyber defence technology," *Procedia Comput. Sci.*, vol. 57, pp. 710–715, Jan. 2015.

[22] M. Denis, C. Zena, and T. Hayajneh, "Penetration testing: Concepts, attack methods, and defense strategies," in *Proc. IEEE Long Island Syst., Appl. Technol. Conf. (LISAT)*, Apr. 2016, pp. 1–6.

[23] B. Arkin, S. Stender, and G. McGraw, "Software penetration testing," *IEEE Secur. Privacy*, vol. 3, no. 1, pp. 84–87, Jan. 2005.

[24] G. Chu and A. Lisitsa, "Penetration testing for Internet of Things and its automation," in *Proc. IEEE 20th Int. Conf. High Perform. Comput. Commun. IEEE 16th Int. Conf. Smart City IEEE 4th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Jun. 2018, pp. 1479–1484.

[25] D. Geer and J. Harthorne, "Penetration testing: A duet," in *Proc. 18th Annu. Comput. Secur. Appl. Conf.*, Las Vegas, NV, USA, Dec. 2002, pp. 185–195.

[26] P. Xiong and L. Peyton, "A model-driven penetration test framework for Web applications," in *Proc. 8th Annu. Int. Conf. Privacy, Secur. Trust (PST)*, Ottawa, ON, Canada, Aug. 2010, pp. 173–180.

[27] W. Xu, B. Groves, and W. Kwok, "Penetration testing on cloud—Case study with owncloud," *Global J. Inf. Technol.*, vol. 5, no. 2, pp. 87–94, 2016.

[28] J. N. Goel, M. H. Asghar, V. Kumar, and S. K. Pandey, "Ensemble based approach to increase vulnerability assessment and penetration testing accuracy," in *Proc. Int. Conf. Innov. Challenges Cyber Secur. (ICICCS-INBUSH)*, Feb. 2016, pp. 330–335.

[29] J. Zhao, W. Shang, M. Wan, and P. Zeng, "Penetration testing automation assessment method based on rule tree," in *Proc. IEEE Int. Conf. Cyber Technol. Automat., Control, Intell. Syst. (CYBER)*, Shenyang, China, Jun. 2015, pp. 1829–1833.

[30] R. LaBarge and T. McGuire, "Cloud penetration testing," *Int. J. Cloud Comput., Services Archit.*, vol. 2, pp. 43–62, Jan. 2013.

[31] V. B. Livshits and M. S. Lam, "Finding security vulnerabilities in Java applications with static analysis," in *Proc. USENIX Secur. Symp.*, Jul. 2005, p. 18.

[32] M. Ceccato and R. Scandariato, "Static analysis and penetration testing from the perspective of maintenance teams," in *Proc. 10th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, Sep. 2016, p. 25.

[33] A. S. Al-Ahmad and H. Kahtan, "Test case selection for penetration testing in mobile cloud computing applications: A proposed technique," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 13, pp. 1–11, Jul. 2018.

[34] T. P. Ping, H. Sharbini, W. B. Lin, V. T. W., and A. A. Julaihi, "Designing a mobile application testing model," in *Proc. Int. Conf. Comput., Netw. Digit. Technol. (ICCNDT)*, Bahrain, Bahrain, Nov. 2012, pp. 255–260.

[35] D. Amalfitano, A. R. Fasolino, P. Tramontana, and N. Amatucci, "Considering context events in event-based testing of mobile applications," in *Proc. Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Luxembourg, Mar. 2013, pp. 126–133.

[36] A. A.-S. Ahmad, P. Brereton, and P. Andras, "A systematic mapping study of empirical studies on software cloud testing methods," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Jul. 2017, pp. 555–562.

[37] T. Siddiqui and R. Ahmad, "A review on software testing approaches for cloud applications," *Perspect. Sci.*, vol. 8, pp. 689–691, Sep. 2016.

[38] D. Anitha and M. V. Srinath, "A review on software testing framework in cloud computing," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 6, pp. 7553–7562, 2014.

[39] H. M. Z. Al Shebli and B. D. Beheshti, "A study on penetration testing process and tools," in *Proc. IEEE Long Island Syst., Appl. Technol. Conf. (LISAT)*, May 2018, pp. 1–7.

[40] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering version 2.3," *Engineering*, vol. 45, no. 4, p. 1051, 2007.

[41] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *Proc. 28th Int. Conf. Softw. Eng.*, Shanghai, China, May 2006, pp. 1051–1052.

[42] H. Zhang and M. AliBabar, "On searching relevant studies in software engineering," in *Proc. 14th Int. Conf. Eval. Assessment Softw. Eng. (EASE)*. Keele, U.K.: BCS, Apr. 2010.

[43] F. Hujainah, R. B. A. Bakar, B. Al-Haimi, and A. B. Nasser, "Analyzing requirement prioritization techniques based on the used aspects," *Res. J. Appl. Sci.*, vol. 11, no. 6, pp. 327–332, 2016.

[44] F. Hujainah, R. B. A. Bakar, M. A. Abdulgabber, and K. Z. Zamli, "Software requirements prioritisation: A systematic literature review on significance, stakeholders, techniques and challenges," *IEEE Access*, vol. 6, pp. 71497–71523, 2018.

[45] W. Du and A. P. Mathur, "Vulnerability testing of software system using fault injection," Purdue Univ., West Lafayette, Indiana, Tech. Rep. COAST TR, 1998, pp. 2–98.

[46] W. Du and A. P. Mathur, "Testing for software vulnerability using environment perturbation," *Qual. Rel. Eng. Int.*, vol. 18, no. 3, pp. 261–272, May 2002.

[47] R. B. Vaughn, R. Henning, and A. Siraj, "Information assurance measures and metrics-state of practice and proposed taxonomy," in *Proc. 36th Annu. Hawaii Int. Conf. Syst. Sci.*, Jan. 2003, p. 10.

[48] S. Singh, J. Lyons, and D. M. Nicol, "Fast model-based penetration testing," in *Proc. Winter Simulation Conf.*, Washington, DC, USA, Dec. 2004, pp. 309–317.

[49] B.-H. Kang, "About effective penetration testing methodology," *J. Secur. Eng.*, vol. 5, no. 5, pp. 425–432, 2008.

[50] Z.-F. Liu, B. Liu, and X.-P. Gao, "SOA based mobile application software test framework," in *Proc. 8th Int. Conf. Rel., Maintainability Safety (ICRMS)*, Chengdu, China, Jul. 2009, pp. 765–769.

[51] P. Xiong, B. Stepien, and L. Peyton, "Model-based penetration test framework for Web applications using TTCN-3," in *E-Technologies, Innovation Open World*. Ottawa, ON, Canada: Springer, 2009, pp. 141–154.

[52] Z. Liu, X. Gao, and X. Long, "Adaptive random testing of mobile application," in *Proc. 2nd Int. Conf. Comput. Eng. Technol. (ICCET)*, Chengdu, China, vol. 2, Apr. 2010, pp. V2-297–V2-301.

[53] H.-C. Li, P.-H. Liang, J.-M. Yang, and S.-J. Chen, "Analysis on cloud-based security vulnerability assessment," in *Proc. IEEE Int. Conf. E-Bus. Eng. (ICEBE)*, Shanghai, China, Nov. 2010, pp. 490–494.

[54] Z. M. Jiang, "Automated analysis of load testing results," in *Proc. 19th Int. Symp. Softw. Test. Anal.*, Trento, Italy, Jul. 2010, pp. 143–146.

[55] T. Vengattaraman, P. Dhavachelvan, and R. Baskaran, "A model of cloud based application environment for software testing," 2010, vol. 7, pp. 257–260, *arXiv:1004.1773*. [Online]. Available: https://arxiv.org/abs/1004.1773

[56] L. Yu, W.-T. Tsai, X. Chen, L. Liu, Y. Zhao, and L. Tang, "Testing as a service over cloud," in *Proc. 5th IEEE Int. Symp. Service Oriented Syst. Eng. (SOSE)*, Nanjing, China, Jun. 2010, pp. 181–188.

[57] S. Baride and K. Dutta, "A cloud based software testing paradigm for mobile applications," *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 3, pp. 1–4, May 2011.

[58] N. Kumar and M. E. Ul Haq, "Penetration testing of Android-based smartphones," M.S. thesis, Dept. Comput. Sci. Eng. Göteborgs Universitetsbibliotek, Gothenburg, Sweden, 2011.

[59] Y. Ridene and F. Barbier, "A model-driven approach for automating mobile applications testing," in *Proc. 5th Eur. Conf. Softw. Archit., Companion*, Essen, Germany, Sep. 2011, p. 9.

[60] W. Jenkins, S. Vilkomir, P. Sharma, and G. Pirocanac, "Framework for testing cloud platforms and infrastructures," in *Proc. Int. Conf. Cloud Service Comput. (CSC)*, Hong Kong, Dec. 2011, pp. 134–140.

[61] T. Wei, J.-F. Yang, J. Xu, and G.-N. Si, "Attack model based penetration test for SQL injection vulnerability," in *Proc. IEEE 36th Annu. Comput. Softw. Appl. Conf. Workshops (COMPSACW)*, Izmir, Turkey, Jul. 2012, pp. 589–594.

[62] P. Kamongi, S. Kotikela, K. Kavi, M. Gomathisankaran, and A. Singhal, "VULCAN: Vulnerability assessment framework for cloud computing," in *Proc. 7th Int. Conf. Softw. Secur. Rel.*, Gaithersburg, MD, USA, Jun. 2013, pp. 218–226.

[63] K. Deptula, "Automation of cyber penetration testing using the detect, identify, predict, react intelligence automation model," M.S. thesis, Naval Postgraduate School, Monterey, CA, USA, 2013.

[64] F. Stahl and J. Ströher. (Sep. 7, 2013). *OWASP and AppSec Security Testing Guidelines for Mobile Apps*. [Online]. Available: https://www.owasp.org/images/0/04/Security_Testing_Guidelines_for_mobile_Apps_-_Florian_Stahl%2BJohannes_Stroeher.pdf

[65] C. Blackwell, "Towards a penetration testing framework using attack patterns," in *Cyberpatterns*. Springer, 2014, pp. 135–148.

[66] M. I. P. Salas, M. Invert, and E. Martins, "A black-box approach to detect vulnerabilities in Web services using penetration testing," *IEEE Latin Amer. Trans.*, vol. 13, no. 3, pp. 707–712, Mar. 2015.

[67] D. S. Cruzes, M. Felderer, T. D. Oyetoyan, M. Gander, and I. Pekaric, "How is security testing done in agile teams? A cross-case analysis of four software teams," in *Proc. Int. Conf. Agile Softw. Develop.*, May 2017, pp. 201–216.

[68] D. He, S. Chan, and M. Guizani, "Mobile application security: Malware threats and defenses," *IEEE Wireless Commun.*, vol. 22, no. 1, pp. 138–144, Feb. 2015.

[69] I. Yaqoob, S. A. Hussain, S. Mamoon, N. Naseer, J. Akram, and A. Ur Rehman, "Penetration testing and vulnerability assessment," *J. Netw. Commun. Emerg. Technol. (JNCET)*, vol. 7, no. 8, pp. 10–18, Aug. 2017. [Online]. Available: https//www.jncet.org

[70] R. Li, D. Abendroth, X. Lin, Y. Guo, H.-W. Baek, E. Eide, R. Ricci, and J. Van der Merwe, "Potassium: Penetration testing as a service," in *Proc. 6th ACM Symp. Cloud Comput.*, Aug. 2015, pp. 30–42.

[71] M. Aksu and C. Li. (2012). *World Quality Report-Mobile Testing: Behind the Curve*. Accessed: Jul. 1, 2014. [Online]. Available: http://www.uk.sogeti.com/Documents/2012-13

[72] S. L. Bangare, S. Borse, P. S. Bangare, and S. Nandedkar, "Automated Api testing approach," *Int. J. Eng. Sci. Technol.*, vol. 4, no. 2, pp. 673–676, 2012.

[73] V. B. Mohata, D. M. Dakhane, and R. L. Pardhi, "Cloud based testing: Need of testing in cloud platforms," *Int. J. Appl. Innov. Eng. Manage*, vol. 2, pp. 369–373, Mar. 2013.

[74] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: Enabling mobile phones as interfaces to cloud applications," in *Proc. ACM/IFIP/USENIX 10th Int. Conf. Middleware*. Cham, Switzerland: Springer, 2009, pp. 83–102.

[75] E. Halash, "Mobile cloud computing: Case studies," M.S. thesis, Elect. Comput. Eng., Wayne State Univ., Detroit, Michigan, 2010.

[76] D. Huang and H. Wu, *Mobile Cloud Computing: Foundations and Service Models*. Burlington, MA, USA: Morgan Kaufmann, 2017.

[77] Y. S. Pundhir, "Cloud computing applications and their testing methodology," *Bookman Int. J. Softw. Eng.*, vol. 2, no. 1, pp. 1–4, Mar. 2013.

[78] D. Dagar and A. Gupta. (2010). *New Paradigm in Conventional Software Testing: Cloud Testing*. Accessed: Feb. 2, 2015. [Online]. Available: http://www.annemary.org/journal/IJITT/New%20paradigm%20in%20conventional%20software%20testing.docx

[79] G. Wassermann, D. Yu, A. Chander, D. Dhurjati, H. Inamura, and Z. Su, "Dynamic test input generation for Web applications," in *Proc. Int. Symp. Softw. Test. Anal.*, Seattle, WA, USA, Jul. 2008, pp. 249–260.

[80] Y. Singh, A. Kaur, B. Suri, and S. Singhal, "Systematic literature review on regression test prioritization techniques," *Informatica*, vol. 36, no. 4, pp. 379–408, 2012.

[81] W. G. J. Halfond, S. R. Choudhary, and A. Orso, "Penetration testing with improved input vector identification," in *Proc. Int. Conf. Softw. Test. Verification Validation*, Denver, CO, USA, Apr. 2009, pp. 346–355.

[82] G. Jones, "Penetrating the cloud," *Netw. Secur.*, vol. 2013, no. 2, pp. 5–7, Feb. 2013.

[83] C. T. Wai. (2002). *Conducting a Penetration Test on an Organization*. Accessed: Dec. 14, 2014. [Online]. Available: http://www.sans.org/reading-room/whitepapers/auditing/conducting-penetration-test-organization-67

[84] G. Weidman. (2012). *Introducing the Smartphone Penetration Testing Framework*. Accessed: Sep. 1, 2014. [Online]. Available: https://media.blackhat.com/ad-12/Weidman/bh-ad-12-smartphone-penetration-Weidman-WP.pdf

[85] N. F. Awang and A. A. Manaf, "Detecting vulnerabilities in Web applications using automated black box and manual penetration testing," in *Advances in Security of Information and Communication Networks*. Cairo, Egypt: Springer, 2013, pp. 230–239.

[86] M. I. P. Salas and E. Martins, "Security testing methodology for vulnerabilities detection of XSS in Web services and WS-security," *Electron. Notes Theor. Comput. Sci.*, vol. 302, pp. 133–154, Feb. 2014.

[87] OISSGroup. (2006). *Information Systems Security Assessment Framework*. Accessed : Apr. 12, 2014. [Online]. Available: http://www.oissg.org/issaf.html

[88] K. Kumar and S. Rao, "A latest approach to cyber security analysis using vulnerability assessment and penetration testing," *Int. J. Emerg. Res. Manage. Technol.*, vol. 3, no. 4, p. 21, 2014.

[89] S. Shah, "Vulnerability assessment and penetration testing (VAPT) techniques for Cyber defence," presented at the Nat. Conf. Adv. Comput., Netw. Secur. (IET-NCACNS), Nanded, India, 2013.

[90] P. Ami and A. Hasan, "Seven phrase penetration testing model," *Int. J. Comput. Appl.*, vol. 59, no. 5, pp. 16–20, 2012.

[91] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, "Technical guide to information security testing and assessment," *NIST Special Publication*, vol. 800, no. 115, pp. 2–25, Sep. 2008.

[92] M. de Graaf, "Intelligent fuzzing of Web applications," M.S. thesis, Software Eng., Univ. Amsterdam Universiteit van Amsterdam, Amsterdam, The Netherlands, 2009.

[93] I. Färnlycke, "An approach to automating mobile application testing on Symbian Smartphones: Functional testing through log file analysis of test cases developed from use cases," M.S. thesis, Dept. Commun. Syst., KTH Royal Inst. Technol., Stockholm, Sweden, 2013.

[94] M. Karami, M. Elsabagh, P. Najafiborazjani, and A. Stavrou, "Behavioral analysis of Android applications using automated instrumentation," in *Proc. IEEE 7th Int. Conf. Softw. Secur. Rel. Companion*, Gaithersburg, MD, USA, Jun. 2013, pp. 182–187.

[95] M. Schneider, J. Großmann, I. Schieferdecker, and A. Pietschker, "Online model-based behavioral fuzzing," in *Proc. IEEE 6th Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Luxembourg, Europe, Mar. 2013, pp. 469–475.

[96] M. Schneider, J. Großmann, N. Tcholtchev, I. Schieferdecker, and A. Pietschker, *Behavioral Fuzzing Operators for UML Sequence Diagrams*. Berlin, Germany: Springer, 2013.

[97] A. Lawanna, "A model for test case selection in the software-development life cycle," *Int. J. Comput., Inf. Sci. Eng.*, vol. 7, no. 4, p. 5, 2013.
[98] M. Raengkla and T. Suwannasart, "A test case selection from using use case description changes," in *Proc. Int. Multi Conf. Eng. Comput. Scientists*, Hong Kong, 2013, pp. 13–15.
[99] H. Hemmati, L. Briand, A. Arcuri, and S. Ali, "An enhanced test case selection approach for model-based testing: An industrial case study," in *Proc. 18th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, Santa Fe, NM, USA, Nov. 2010, pp. 267–276.
[100] B. R. Chang, H. F. Tsai, Z.-Y. Lin, and C.-M. Chen, "Access security on cloud computing implemented in Hadoop system," in *Proc. 5th Int. Conf. Genetic Evol. Comput. (ICGEC)*, Kitakyushu, Japan, Sep. 2011, pp. 77–80.
[101] G. de los Reyes, S. Macwan, D. Chawla, and C. Serban, "Securing the mobile enterprise with network-based security and cloud computing," in *Proc. IEEE Sarnoff Symp.*, Newark, NJ, USA, May 2012, pp. 1–5.
[102] T. Kim, Y. Choi, S. Han, J. Y. Chung, J. Hyun, J. Li, and A. W.-K. Hong, "Monitoring and detecting abnormal behavior in mobile cloud infrastructure," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, Maui, HI, USA, Apr. 2012, pp. 1303–1310.
[103] L. Q. Sumter, "Cloud computing: Security risk," in *Proc. 48th Annu. Southeast Regional Conf.*, Oxford, MS, USA, 2010, p. 112.
[104] Z. Zhou and D. Huang, "Efficient and secure data storage operations for mobile cloud computing," in *Proc. 8th Int. Conf. Netw. Service Manage.*, Las Vigas, NV, USA, Oct. 2012, pp. 37–45.
[105] S. Bahl and M. M. Chaturvedi, "Literature review of mobile applications testing on cloud from information security perspective," *Int. J. Comput. Appl.*, vol. 79, no. 14, pp. 15–23, Jan. 2013.
[106] O. Starov, "Cloud platform for research crowdsourcing in mobile testing," M.S. thesis, Softw. Eng., East Carolina Univ., Greenville, NC, USA, 2013.

AHMAD SALAH AL-AHMAD received the B.Sc. and M.Sc. degrees in computer information technology from Yarmouk University, Jordan, and the Ph.D. degree in information technology and quantitative sciences from Universiti Teknologi MARA, Malaysia. He is currently an Assistant Professor with the American University of the Middle East, Kuwait. His research interests include data security, mobile technology, and cloud computing applications.

HASAN KAHTAN received the B.Sc. degree from the University of Baghdad and the M.Sc. and Ph.D. degrees in computer science from Universiti Teknologi MARA, Malaysia, in software engineering and software security. He is currently working as a Senior Lecturer with the Faculty of Computing, Universiti Malaysia Pahang. His research interests include software engineering, software security and dependability attributes, and machine learning.

FADHL HUJAINAH received the B.Sc. degree (Hons.) in computer science and software engineering and the M.Sc. degree (Hons.) in information technology from Universiti Teknologi Malaysia, Johor Bahru, Malaysia, in 2012 and 2013, respectively, and the Ph.D. degree in software engineering from University Malaysia Pahang, Kuantan, Malaysia, in 2019. He is currently a Senior Lecturer with the Faculty of Computing, University Malaysia Pahang. His research interests include software engineering with particular interest in software development, requirements prioritization, stakeholder analysis, and decision making.

HAMID A. JALAB received the B.S. degree in electrical engineering from the University of Technology, Iraq, and the M.Sc. and Ph.D. degrees in computer systems from Odessa National Polytechnic University. He is currently an Associate Professor with the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. He has authored more than 100 indexed articles and ten national research projects. His research interests include digital image processing and computer vision.