IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 16, 2021, accepted June 1, 2021, date of publication June 10, 2021, date of current version June 24, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3088229*

# Penetration Frameworks and Development Issues in Secure Mobile Application Development: A Systematic Literature Review

**IKRAM UL HAQ AND TAMIM AHMED KHAN**
Department of Software Engineering, Bahria University, Islamabad 44000, Pakistan
Corresponding author: Tamim Ahmed Khan (tamim@bahria.edu.pk)

**ABSTRACT** The invention of smartphones has opened a new market for mobile application development. Amateur android app developers often do not possess knowledge of the latest android vulnerabilities and thus create applications with attack surface that hackers exploit. In this literature review, many available frameworks and techniques have been analyzed using ISO/IEC 25010 software quality model and identified challenges that android developers face in designing a secure application for android. This paper also presents a comprehensive survey of different penetration tools, evaluated by using criteria such as code analysis, code review, vulnerability analysis, vulnerability exploit, payload and whether these can be used in vulnerability modeling during the design phase. Our study effectively identifies the issues and gaps which can further help develop a framework/tool for designing a penetration secure mobile application by embedding all the vulnerabilities during the design phase using an android vulnerability repository.

**INDEX TERMS** Android, penetration testing, android security and privacy.

## I. INTRODUCTION

Android is an open-source, Linux kernel-based mobile operating system. It allows developers to develop native applications using the Android SDK. According to Gartner Inc. [29], the sales stats of 2017 showed that 86.1% of mobile phones users are using Android OS. There are ten different android operating systems and each version of android OS API vary in functionality and security. The latest API 29 was used in version 10.Android applications are in the form of an APK package. When we decompress an APK package we get the following files and folders:-

- AndroidManifest.xml: This file contains permissions, configuration and security details of each of the component.
- resources.arsc: This file contains compiled resources.
- classes.dex: This file executes when the application runs. It contains the Dalvik Bytecode.
- Res: This folder consists of images etc.
- Assets: This folder contains a database, videos etc.

Android has its own security mechanism and multiple applications utilize a shared memory space. Despite sandboxing and implementation of security models, android is prone to vulnerabilities with increased number of malware attacks [27]. A report showed that 60 games on google play store were malware-infected [39]. The system resources can be accessed by an application if the users assign permissions at the time of installation of the application [70].

The Android architecture, consist of the following components:-

- Activities: It provides an interface where a user can perform an activity such as making a call, sending SMS and so on.
- Services: It runs in the background and performs long terms desired functionality such as alarm app running in the background.
- Broadcast receivers: It performs a system vide operation announcements.
- Content providers: It presents data to external applications. The application can share data and functionality using content providers.

Android has a WebKit engine-based browser and for data storage, it has an SQL database engine (SQLite) [28]. User can download and install various ''third party'' applications from Google Play [70]. There are three types of Android applications: (1) Native Applications are Platform specific

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

applications. These applications can have access to camera, accelerometer, SMS, Contacts, etc. (2) Web applications look like native application but they are in fact HTML coded pages loaded in a browser window. These apps do not have access to the system features like camera, contact etc. (3) Hybrid Applications have both properties of native as well as web apps. A part of these app utilizes browser while the rest can use the system features.

A number of applications being developed for Android OS are prone to threats and can act maliciously. These applications can affect the users in a number of ways, from draining the battery to the theft and loss of user's personal data. A safe application can become malicious by acquiring unnecessary permissions from the user at the time of installation. An application can (1) send paid SMS (2) steal personal data (3) dial calls (4) initiate harmful activities using your mobile phone as bait. Permission Gap occurs when an additional set of permissions is acquired by the application which are not mandatory. This Permission Gap helps the malware to achieve goals like code- injection [10].

Android applications can interact with each other and can pass messages to provide services to each other through android message passing system. Malwares such as viruses, Trojans and spy tools pose a serious threat to mobile devices such as by leaking personal information, causing financial loss, depleting battery power, and degrading network performance. It is important to understand that the entry points in the application can pose a threat to the user data if not handled properly [18]. Therefore, the changing Android ecosystem impacts the app performance adversely [42].

Penetration testing, also known as Pen-test, helps in identifying vulnerabilities in an application using test scenarios to uncover issues that can allow an intruder to gain access to a data/system. Pen-testing performed at the time of deployment of the application and is not executed at each stage of the software development life cycle. Post-deployment security fixes, lacking of regression testing, produce ancillary errors and application can become vulnerable. Lack of pen-testing by mobile application developers also result in insecure mobile applications [5].

Unencrypted data in the banking applications leads to Man-in-the-Middle (MitM) attacks. Enterprises, financial institutions, banks and third-party laboratories should perform vulnerability assessment in detail once a year or twice a year for security inspection of mobile banking applications for identification of vulnerabilities and exploits [126]. Ethical hackers can perform penetration testing to gain access to a system/network and exploit the vulnerabilities in the system [16]. This type of hacking is considered legal because vulnerabilities are reported to the owner of the system so that improvements can be made to the system from a security point of view [40]. Deficiencies in the system network and applications can help the hackers in entering the system where they can damage the system [24]. The security and reliability (sr) of the application is dependent and directly proportional to the expertise (e) of the tester ($sr \propto e$). Many pen-test methodologies exist where different tools and techniques are employed but each methodology starts after a module or application is deployed. The pen-test phases involve gathering application information, vulnerability analysis, and exploitation [7].

From a secure application development point of view, it is important that the application designer must have a complete knowledge of the threat model for the possible attack surface and threats to the entire application architecture. Lack of developer's experience and security knowledge of different vulnerabilities often result in vulnerable applications [58] [5]. Threats and malware often appear due to excessive /mishandled permissions [71]. Lack of best programming practices while programming an android application can lead to a malicious application [17]. Security implementations during software development receive less consideration and attention, in an effort to save time and cost [67].

The solution to complex problems can be acquired through Model-Driven Development (MDD) [8]. Various modelling languages may be used to design abstract models that have significant impacts by translation of model to code using various tools. Unified Modeling Language (UML), for instance, is used in the software application development domain which consists of a set of notations that can be utilised to model structural and behavioural aspects of the systems [48]. Nowadays, although UML is commonly used to model android based applications, but no notations and stereotypes are deployed in UML that can effectively help in modelling security aspects in applications for Android OS [76], [77].

In this paper, we investigated tools and frameworks which can help to model threats during mobile app development. Additionally, we explored the issues developers face during android application development and made the following contributions:

1. We discussed android vulnerabilities and how they affect users' privacy that can further lead to information theft as well as resources leakage. The problem originates when amateur android developers do not take proper security measure due to lack of android security knowledge which hackers exploit through SQL Injections, gaining privileges, malware penetration for gaining access to device. To the best of our knowledge, this is the first work that discusses the android development challenges and solutions in relation with existing penetration frameworks and tools.
2. We also reviewed existing android security framework and penetration tools through systematic literature research. We evaluated existing penetration test tools and frameworks to figure out whether they can help an amateur designer/developer who do not have android OS security knowledge. We also discussed challenges faced by developers while developing android applications.

A brief background and related work on android OS and vulnerabilities is given in Sect. 2. In Sect. 3, we presented

systematic literature review. Section 4 presents the results of systematic literature review and answers to research questions. Finally, Sect. 5 concludes the paper.

## II. BACKGROUND AND RELATED WORK

As discussed earlier, the three types of mobile applications (Native applications, Web Applications and Hybrid Applications) possess different attack surfaces and threat models. While developing an application, the developer has a responsibility to consider the safety of users' data as well as threat models and other constraints such as battery usage, memory constraints and securing possible entry points for gaining access to mobile applications [15]. Excessive permissions of an application can be vulnerable. Such permissions are granted at the time of app installation [63]. Security implementations during software development receive less consideration and attention. For assuring security in web application development the emphasis should be given on security checks implementation during the entire web development life cycle [5]. Shuaibu *et al.* [67] explain that 39 secure frameworks were studied and only 3 percent of the framework had security incorporated.

Tondel *et al.* [74] surveyed various approaches regarding requirement elicitation from a security point of view. Developers do not consider security during the development of application and mainly focus on the completion of functionality. Requirement engineering phase does not focus on security objectives and identifying threats. Microsoft Security Development Lifecycle [1] is used for developing a secure application by integrating security in all phases of software development but it does not focus on how penetration testing should be performed at various stages of SDLC. SDLC is generic and is not android specific. Xiong *et al.* [82] proposed a web penetration framework using the grey-box approach which could be incorporated in SDLC but it does not help in securing android application during design and development. Denis *et al.* [24] explained different methods, tools and techniques of penetration testing and performed tests on smartphones. The attack results showed that 14.6% issues were due to the lack of application security hardening. It was also mentioned that despite Google play not allowing malicious applications, there are still applications that carry malware or malicious code.

Unified Modeling Language (UML) helps in modelling the artefacts for a system [32]. Eoin *et al.* [81] explained the limitations of UML as it does not have specific elements to model system interaction. However, UML profile can be enhanced for specific modelling so that the system can be modelled accordingly. Goel *et al.* [31] presented a new methodology for secure modelling but it does not have any information on how to model a secure application for Android. Minhyuk Ko *et al.* [46] extended the UML and defined new elements so that the relationship of entities can be defined. Bup- ki Min *et al.* [46] presented new profiles for Windows 7 phones and Android OS respectively but none of these profiles cover the security aspects on how permissions can be handled and secure IPC (inter-process communication) can be modelled. Bo *et al.* [13] presented mobile test using event-based approach for the automation of test cases. This tool does not help in identifying vulnerabilities and producing test case artefacts during development.

Although penetration testing is gaining importance since the last few years, there are few systematic literature reviews available regarding penetration in android based applications. Mirjalili and Alidoosti [54] reviewed the penetration methodologies and vulnerability scanning tools for web-based applications. The review was based on 4 different web penetration methodologies and 13 different vulnerability testing scanners using evaluation criteria. The author further reviewed four test environments (Webgoat, DVWA, BodgeIt and WackoPicko). Based on the evaluation the author concluded that there are many vulnerabilities which existing tools are unable to identify.

Shanley and Johnstone [64] reviewed six penetration test frameworks and methodologies. The author mapped ISO/IEC 25010:2013 quality model as evaluation criteria on two of the methodologies (OWASP's OTG and ISSAF) and found that these frameworks lack domain coverage and have restrictions. Bertoglio and Zorzo [11] performed a systematic mapping study on a selection of 1145 papers. A detailed assessment of penetration testing methodologies, tools, and models was performed using evaluation criteria in a quantitative and qualitative way. The author concluded that how these tools and techniques may be applied for vulnerability assessment and what is the limitation of various models used in penetration testing for a web-based application.

Shah and Mehtre [126] deliberated that in Penetration testing, potential entry points into a device are identified using standard hacking tools and techniques. Different hacking tools were discussed along with comparative analysis of techniques and methodologies and there implementation on Banking apps to identify vulnerabilities. Felderer *et al.* [53] examined various model-based testing techniques and reviewed 119 publications and concluded how coverage criteria can impact the feasibility and return on investment. Al-Ghamdi [2] reviewed various software security techniques and how flaws during software developments can be exploited. The author concluded that the software developers should take security measures by security validation and remove security flaws for the effectiveness of the software application.

Xiao, Liang, *et al.* [125] proposed malware detection in mobile phones using Q-learning. The detection performance is improved using Dyna architecture and reinforcement learning process. However, the paper does not discuss how Q-learning can be used to improve detection using vulnerability repository during application development.

### A. ANDROID THREATS CLASSIFICATIONS

Android's popularity is increasing and so is the android malware being developed more frequently. A huge number of

**TABLE 1.** Vulnerabilities detected in 2018–2019.

| Year | Android version | Number of vulnerabilities | Type of vulnerability |
|---|---|---|---|
| 2018 | 9 | 362 | SQL Injection, Access Control, remote code execution, denial of service, Memory leak, buffer overflows, code Bypass, Exec code overflow, the elevation of privileges. |
| 2019 | 10 | 193 | Remote code execution, DoS (denial of service), Elevation of privilege, Information disclosure, SQL injection, exec code bypass. |

attack vectors and vulnerabilities are detected in the Android OS [16]. Table 1 shows security vulnerabilities reported in 2018 and 2019 in android OS. Major android OS security vulnerabilities which hacker's exploit are:

- DoS
- Code Execution
- Overflow
- Memory Corruption
- SQL Injection
- XSS
- Directory Traversal
- Http Response Splitting
- Bypass something
- Gain Information
- Gain Privileges
- CSRF
- File Inclusion

Apart from these security vulnerabilities, there are many applications which provide attack surface through their behaviour. These applications include:-

- Spyware Applications which transmit data to the unauthorized destination as well as captures the user's behaviour.
- MUS (Mobile Unwanted Software) which collects information about files, accounts and other device information without user authorization.
- Backdoor applications which allow remote operations and allow access to the device.
- Downloader Applications display security alerts and force users to download malicious applications.
- Call, SMS, Toll fraud Applications make unauthorized call or SMS by misleading and tricking users and charge users.

Figure 1 shows a graph of android OS vulnerabilities for the year 2016-2019, whereas vulnerability details are shown in Figure 2.

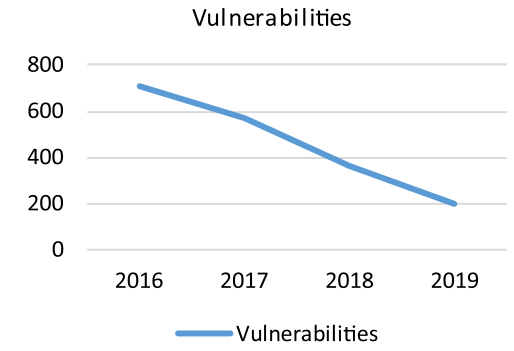



**FIGURE 1.** vulnerabilities for the year 2016–2019.

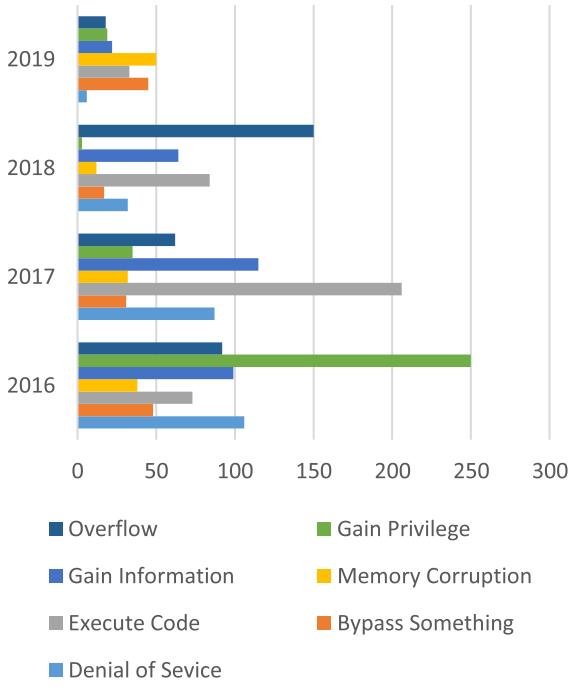



**FIGURE 2.** vulnerability details.

## III. REVIEW PROCESS

During systematic literature review (SLR) process, we will be identifying, evaluating and interpreting available research papers which relate to our research question and topic area. We will summarize existing researches for identifying gaps in the prevailing literature for positioning our research. We will follow guidelines by Kitchenham *et al.* [14] as shown in figure 3.

### A. RESEARCH QUESTIONS

In this paper, we present the results of systematic literature review (SLR) on the topic of model-driven penetration test framework and tools. We will be reviewing the literature of the software engineering and software security field. In the review, the following research questions (RQ) will be studied.

**TABLE 2.** Matching keywords.

| Rationalization | Keyword | Other Matching Keyword |
|---|---|---|
| **Population** | Mobile App Security | |
| **Intervention** | Android penetration testing | Android Pen-Test , Android Penetration, Mobile Penetration Testing,Penetration Testing, Pentest |
| **Outcome** | Methodologies, Frameworks, Tools | Mobile security tools, Toolkits,Security issues, Frameworks, Challenges |

**RQ1.** What are the different penetration test frameworks and tools and how they help in developing a secure application?

**RQ2.** What challenges Android app developers encounter and how to meet the secure coding practices in terms of confidentiality and authentication?

We choose only two questions because we believe that these RQ could provide two distinct and actionable insight to the researchers.

### B. SEARCH PROCESS

These research repositories were searched for a combination of keywords i.e. Penetration testing, secure android applications framework, pen-test framework, android developer challenges, pen-test and Penetration test in android. Search keywords were derived from PICO as shown in Table 2.

- Population (P): Mobile Application Security
- Intervention(I): Android penetration testing
- Comparison(C): Existing Pen-test and Security Frameworks
- Outcome(O): Methodologies, Frameworks, tools, challenges

### C. RESEARCH PROCESS

The research process consisted of three main stages as shown and explained in Figure 5. The search process for the selection of research studies was done in two stages, first using keywords-based search from different following repositories and second, by manually filtering them.

- ACM Digital Library
- IEEE Explore
- Google Scholar
- HEC digital library
- Springer link

Identification of research from the above sources was manually filtered by the first author on the basis of inclusion and exclusion criteria. The full text was then analyzed and duplicates were removed. The entire process was mentored and audited by the second author.

### D. INCLUSION AND EXCLUSION CRITERIA

Search results retrieved a large number of results so Inclusion (IC) and Exclusion criteria (EC) is applied to reduce the number of papers and to select the quality research papers. The IC and EC are defined as below.

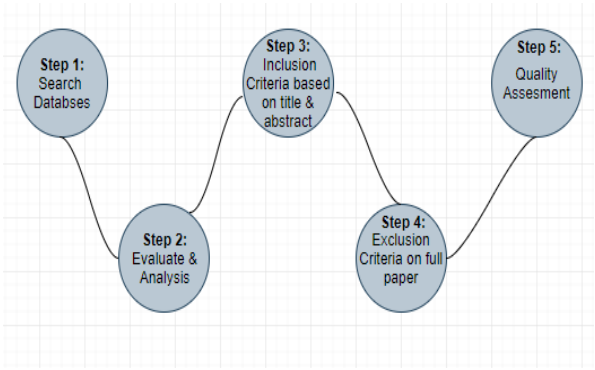



**FIGURE 3.** Selection process.

- IC1- The paper or abstract or introduction matched with the study in question and explains a framework, methodology or tool for penetration testing.
- IC2- The paper discusses penetration testing tools for android applications.
- IC3- Papers written in English and published in Journals, books and conferences
- EC1- Exclusion of all articles that do not answer research questions.
- EC2- Articles that cover penetration testing other than the web and mobile applications

The full text of paper was analyzed and duplicates were removed.

### E. STUDY SELECTION

In the data selection process, initially, the primary studies identified during search were refined by filtering on the basis of abstract and then on the basis of full text. The keyword match search pulled up 1040 results from the search repositories and after removing duplicates and irrelevant papers we were left with 380 papers. The abstract of the paper was categorized by the first author in three categories: Uncertain, include and exclude. The papers in these categories were discussed by both the authors and as auditing by the second reviewer we were left with 56 papers. The resulting research was tabulated to: -

- Show the penetration test frameworks, methodologies and tools (to address RQ1).
- Identify issues and challenges faced by android app developers (to address RQ2)

*(("Android Pen-Test" OR "Android Penetration" OR Mobile Penetration Testing OR Model-based penetration testing "Penetration Testing "OR OR "Pentest") AND (mobile security, software tools, toolkits) AND (methodology OR framework) AND ("Android developers Issue" OR Security issue OR Android challenges OR Pen Secure Android App Design OR Development))*

**FIGURE 4.** Search string.

We used logical operators "AND" and "OR" in search string for selection, shown in Figure 4.

### F. QUALITY ASSESSMENT CRITERIA

To evaluate each study using unbiased strategy and to increase validity for extraction of relevant publications we used qualitative assessment criteria based on [14], [55], [56]. The criteria were based on the following Quality assessment (QA) questions.

QA 1. Is the research relevant to penetration testing and is relevant to the domain?
QA 2. Does the paper use the framework, methodology, tool effectively?
QA 3. Is the framework, methodology, tool discussed in the paper answer the research question?
QA 4. Does the paper cover issues and challenges related to amateur android developers

The quality assessment (QA) questions were scored as follows:

QA 1. Y (yes), the study is relevant to penetration testing; N (no), the study is not relevant to penetration testing; P (Partly), the study is relevant to penetration testing but does not mention its implication on android apps.
QA 2. Y (yes), the study uses a framework, methodology or tool; N (no), the study does not mention a framework, methodology or tool; P (Partly), the study does not provide an evaluation of penetration framework, methodology or tools
QA 3. Y (yes), the study answers the research uestions; N (no), the study does not answer the research questions; P (Partly), the study partly answers the research questions with no validations.
QA 4. Y (yes), the study concludes the challenges; N (no), the study does not conclude the challenges; P (Partly), the study partly answers the challenges related to the tools, methodologies and frameworks.

The scoring procedure for quality assessment questions is Y (yes) = 1, N(no) = 0 and P(partly) = 0.5. Since we have four QA questions so the total score is 4.

**TABLE 3.** Evaluation results.

| Database | Retrieved | Applying Criteria. | Selected |
|---|---|---|---|
| IEEE Xplore | 257 | 96 | 10 |
| ACM DL | 65 | 24 | 7 |
| HEC Digital Library | 310 | 112 | 14 |
| Springer Link | 42 | 16 | 9 |
| Google Scholar | 366 | 132 | 16 |
| **Total** | **1040** | **380** | **56** |

### G. DATA COLLECTION

In the data selection process, all the exclusion and inclusion criteria were implemented on the search results and duplicate papers were removed. Initially, the data was collected using the search criteria which was based on keywords. Further, the data extracted from each study were:

- Main research topic.
- Main research questions and answers
- Quality evaluation (if any)
- Pentest methodologies and tools
- Issues and challenges

### H. DATA ANALYSIS

The collected data was tabulated to: -

- Show the penetration test framework, methodologies and tools (to address RQ1).
- Identify issues and challenges faced by android app developers (to address RQ2).

### I. PAPER SELECTION

Using the search criteria, initial search pulled up 1040 results from the mentioned repositories and after removing duplicates and irrelevant papers we were left with 380 papers. After applying Inclusion and Exclusion criteria and after reviewing the title and reading the abstract we were left with 56 papers as shown in Table 3.

### J. QUALITY ASSESSMENT

Quality of the SLR can not only be assessed using inclusion and exclusion criterias as it may give biased results. Hence, specific quality assurance(QA) criteria are used to fully evaluate the text, thus increasing the selection of lilterature [61]. The desired score for evaluation is 3.5 and the score is categorized on the basis of availability of several parameters as mentioned in research questions.

### K. QUALITY ASSESSMENT RESULTS

The paper shortlisted using the criteria mentioned above were further assessed using the quality assessment criteria as mentioned in section F. The desired score for evaluation is 2.5 and the score is categorized on the basis of availability of several parameters as mentioned in research questions. Not many studies revealed a good score. The result discussed

in Table 4 are obtained after applying evaluation criteria on 56 papers which match the threshold criteria.

### L. THREATS TO VALIDITY

Researcher bias: This can influence the overall research selection and extraction process by selection of inclusion and exclusion criteria falsely. This threat was mitigated by the teamwork of both the researchers. Inclusion and exclusion criteria were designed by both the researchers and all uncertain cases were discussed and resolved mutually. All the cases, where the paper was included or excluded was done on the mutual agreement.

Subjective bias: Subjective bias was mainly related to the limitations in acquiring results through the use of keywords. During the search process, we may have missed papers which were unpublished. Furthermore, our search criteria were purely based on our knowledge and expertise and there could be a possibility that we would have missed some papers that have used synonyms of our search keywords. To mitigate this issue, initially, we tried a formal search with a keyword and its synonym. The results were analyzed and in such case i.e. where we used a keyword ''Android Pen-Test'' we added another keyword ''penetration test'' to make sure that we do not miss any research papers.

## IV. RESULTS AND DISCUSSION

We discuss our results in this section.

**1. Android Penetration test tools and frameworks (RQ1)**

On the basis of our systematic literature review (SLR) and quality assessment criteria, we have grouped the results in two categories: Penetration test frameworks and Penetration test tools. These frameworks and tools were further evaluated on the basis of two separate evaluation criteria.

**2. Penetration test frameworks (RQ1-part 1)**

We have selected below mentioned frameworks from our systematic literature and will further evaluate these frameworks on the basis of our evaluation criteria to answer one portion of our research question.

***a. Open Source Security Testing Methodology Manual (OSSTMM)***

Open Source Security Testing Methodology includes a set of strategies for security and quality that can be implemented for penetration and vulnerability testing. OSSTMM audit covers each aspect and relationship among software, system, people and processes. Through Passive and Intrusive attacks, a review report is generated which helps both, the developers and the network professionals to improve security.

OSSTMM consists of (1) COMSEC (communications security channel) for human and physical interactions (2) SPECSEC (spectrum security channel) for wireless signals and communication (3) PHYSSEC (physical security channel) class for Telecommunications and data networks. OSTMM focuses on testing of application from penetration point of view and what measures should be taken after the tests are performed. This methodology does not cover what tools should be used for evaluation and testing of various interactions and classes [95].

***b. OWASP (Open Web Application Security Project)***

OWASP is a community dedicated to providing security guidelines, tools, secure coding standard (ASVS) and documentation to improve application security. OWASP framework consists of (1) Information Gathering (2) configuration of information (3) logging (4) Session testing (5) authorization testing. OWASP Mobile Application Security Verification Standard helps in embedding security in mobile apps.

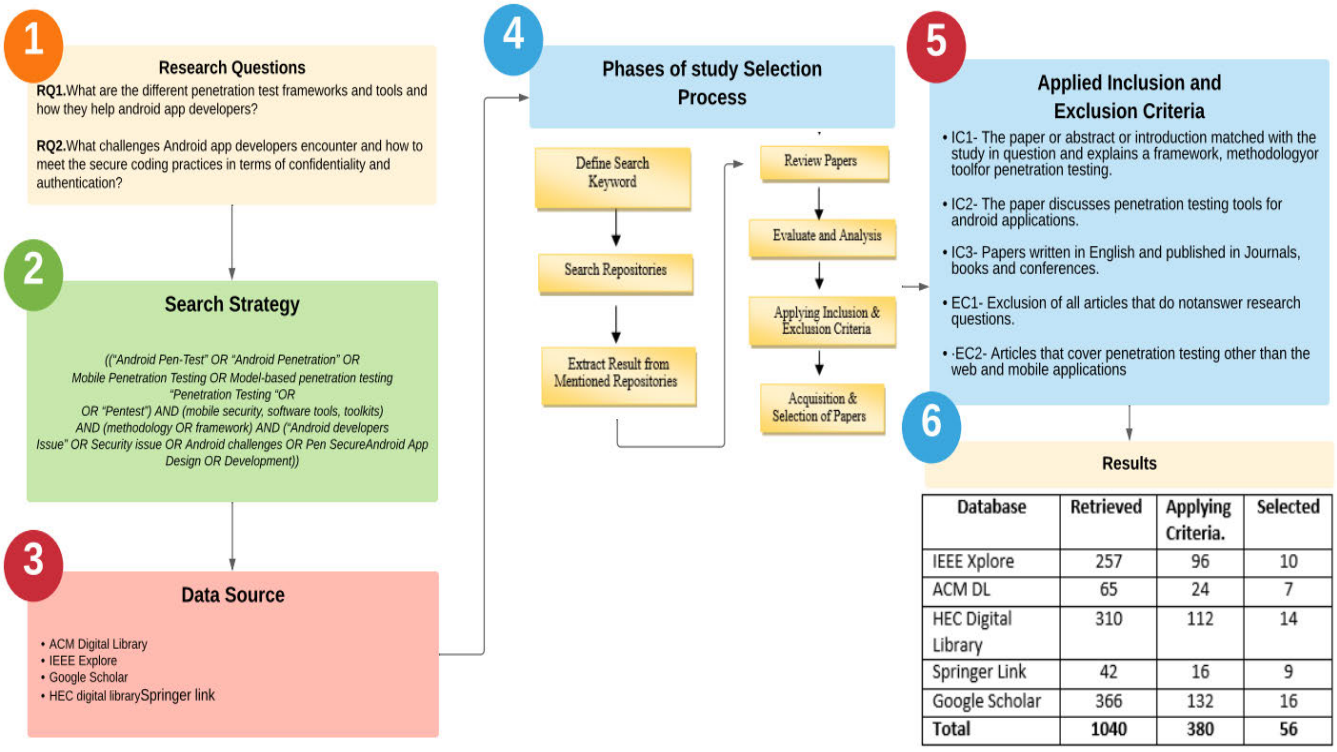
MASVS has two security levels L1 and L2 as well as protection of app from reverse engineering or tampering. Both MASVS-L1 and MASVS-L2 guidelines provide best practices that cover all the aspects of security threats. MASVS define two security levels (L1 & L2) and resiliency requirement (MASV-R) to protect against tempering and reverse engineering. MASVS-L1 covers standard security requirements for mobile applications such as data handling and interaction of app in mobile environment. MASVS-L2 covers in-depth security beyond standard security requirements. App can achieve L2 level if threat model exits and based on the threat model control have been implemented.

MASVS-R signifies discretionary protective layer for obstructing reverse engineering and app tampering. MASVS-R can be applied on apps which handle sensitive data and functionality is applied. The app can have state of art security if it has clearly defined tempering and reverse engineering attacks. To achieve this level, the application can leverage verifiable techniques for software protection and hardware security features.

OWASP identifies top ten serious application security risks and their technical and business. It also provides methods to deal with these security risks. OWASP top ten security checklist for secure programming is a great resource for developers [96]. OWASP provides resources to the developers to develop controls for the security risks and build secure applications [97].

***c. NIST ISAM (Information Security Assessment Methodology)***

NIST Pen-test methodology consists of Four Phases which include (1) Planning (2) Discovery (3) Attack (4) Reporting. During Planning phase objectives are defined, information about the SUT (System under test) is gathered and vulnerability analysis is performed. During execution, various tests are performed to determine vulnerabilities and finally identified vulnerabilities are reported. NIST ISAM focuses on developing policies for penetration test and security assessment. Policies and technical considerations alongwith standardized testing techniques can result in improved security [98]. NIST standard NIST 800-163 [110] provides guidelines for the security vetting of mobile application. The application vetting process is a series of activities to check if the mobile app security conforms to the security standards of the organisation. App vetting is done in following steps:-




| Database | Retrieved | Applying Criteria. | Selected |
|---|---|---|---|
| IEEE Xplore | 257 | 96 | 10 |
| ACM DL | 65 | 24 | 7 |
| HEC Digital Library | 310 | 112 | 14 |
| Springer Link | 42 | 16 | 9 |
| Google Scholar | 366 | 132 | 16 |
| **Total** | **1040** | **380** | **56** |

**FIGURE 5.** Overall SLR process.

- Security Artifacts are reviewed
- App vulnerability testing is done and report is generated (Test tools are used)
- Compliance with organization security standards is checked and recommendations are given.

### *d. Information Systems Security Assessment Framework (ISSAF)*

ISSAF methodology is a peer-reviewed framework designed to assess security and suggest standards for each domain by reflecting actual scenarios. ISSAF consists of activities which include information gathering, network mapping, vulnerability assessment, penetration testing, privilege escalation, enumeration, gaining access and covering the tracks. ISSAF, however, does not provide security guidelines for mobile phone app development.

### *e. PTES (Penetration Testing Execution Standard)*

Penetration Testing Execution Standard includes several other standards and guidelines such as OWASP for web penetration testing for conducting tests in web and other applications. PTES framework consist of (1) Pre-engagement Interactions to define scope (2) Intelligence Gathering to gather information about the target system (3) Threat Modelling for pen-test execution (4) Vulnerability Analysis for discovering vulnerable behavior and flaws in SUT (5) Exploitation for establishing unauthorized access (6) Post Exploitation for covering the tracks (7) Reporting for the customer with suggestions.

## 3. Penetration test frameworks Evaluation

The shortlisted above listed penetration test methodologies and frameworks are evaluated on the basis of following criteria from ISO/IEC 25010:2013 software quality model to check if these frameworks provide any model-based support to the developers/ designer:

- **Coverage:** It refers to the scope of the methodologies to deal with every aspect of pen-testing ranging from a web-based application to mobile native apps.
- **Flexibility**: Flexibility refers to the ability of penetration test methodologies to allow community developers and testers to extend pen-tests capability by adding/extending guidelines.
- **Adaptability:** The framework should have well defined and non-ambiguous guidelines so that testing can be performed on android OS-based apps.
- **Modelling:** Modelling comprehends the realization of MDE (model-driven engineering) and provides Model-based support to embed security in design using prebuild stereotypes and class diagrams.
- **Usability for amateur developers:** The methodology Provides a knowledge base and tools which can help in automatically identifying vulnerabilities during the modelling of application.

## 4. Evaluation Results

OSSTMM 3, NIST-ISAM, ISSAF, and PTES show coverage issues of various application domains from

**TABLE 4.** Literature quality assessment.

| *S.NO* | *YEAR* | *DISCUSSION* | *PROS /CONS* | *QA1* | *QA2* | *QA3* | *QA4* | *TOTAL SCORE* |
|---|---|---|---|---|---|---|---|---|
| *1* | Mathew et. al. [3] | Different aspects of penetration testing tools, techniques and methodologies have been discussed. Vulnerabilities exploitation and hacking of smart phones, blue tooth devices WPA was discussed. | The author explained different types of penetration tests, tools and techniques. However, the paper does not cover how effectively they can be used for development of secure android application. | *Y* | *Y* | *P* | *N* | *2.5* |
| 2 | Petukhov and kozlov [4] | The paper proposes a framework employing dynamic analysis, black box test and static methods together, through which any weakness of one method can be addressed by the strength of the other. The proposed frame work claims to deliver better results by amalgamating the functions of the three methods and than employing each disjointedly. | Vulnerabilities of applications and services are effectively discovered using proposed penetration test framework. The paper does not cover how proposed methods can be employed on android environment. | Y | Y | P | N | 2.5 |
| *3* | Bacudio et al. [7] | The paper reviews the penetration testing, its effectiveness and the methodology. The thorough methodology analysis involves identification of the three methodology phases; test preparation, test and test analysis and illustrates the application of the same on two web applications. | The author discussed the penetration test methodologies and frameworks but the paper lacks discussion on the tools and techniques for pen-testing. | *Y* | *P* | *P* | *N* | *2* |
| *4* | Sicari et al. [12] | Security of middleware, solutions for securing mobile devices and how vulnerabilities can be handled was discussed. | Lack of discussion on penetration test frameworks and developer related challenges for Android app development. | *P* | *Y* | *P* | *N* | *2* |
| 5 | H. et. al. [19] | Penetration testing and related factors were discussed. Various techniques and tools for employing security was elaborated. ISO 27000 security standard as well guidelines and ethics for developers for secure coding were discussed. | The author discussed the penetration test methodologies and frameworks but he paper lacks framework and tools which can help in modeling security vulnerabilities. | Y | Y | P | P | 3 |
| 6 | Mosenia et al. [20] | Author has discussed different security vulnerabilities and their countermeasure. There is an exponential increase in weak link frequency and unexpected use of user's data. | The paper does not cover the how effectively the pen-test framework can be used for development of secure android application and what challenges developers face. | p | p | P | N | 1.5 |
| 7 | Gerry et al. [21] | The paper discusses how the developers are made familiar with the communication protocols such as Ethernet, ARP, IP, ICMP, UDP and TCP protocols. | Lack of discussion on penetration test frameworks. The paper does not cover how proposed methods can be employed on android environment. | P | P | N | N | 1 |
| 8 | Dawson and mcdonald [22] | The paper covers how security loopholes can lead to increased cyber-attacks and how secure software engineering can lower the exploitation of vulnerabilities in an application. The discipline, The paper suggests a model for improving security in the application. | The author and explained different types secure programming techniques. However, the paper does not cover the penetration test frameworks. | p | Y | p | p | 2.5 |
| 9 | Priya et. al. [24] | The author has conducted a survey on different penetration test tools and techniques and discussed how hacking takes place. The paper also discussed various methods for securing the application. 10 | Lack of discussion on penetration test frameworks and developer related challenges for Android app development. | Y | Y | P | P | 3 |

**TABLE 4.** *(Continued.)* **Literature quality assessment.**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 10 | Bing et al. [25] | The paper provides an overview on Pentest, showing its application scenarios, models, methodologies, and tools from published papers. The paper conducts a systematic mapping and presented some open issues and research opportunities on Pentest. | The research contains different types of penetration tests, tools and techniques. However, the paper partially cover the penetration test framework and how effectively they can be used for secure development. | Y | Y | P | N | 2.5 |
| 11 | Xue et.al. [26] | The authors proposed an automated method for resolving development issues like high costing & low efficiency during penetration tests. | The author deliberated methodologies and frameworks for penetration testing but the paper lacks discussion on the tools and techniques for pen-testing. | p | p | P | P | 2 |
| 12 | Lane et al. [35] | The paper proposes CVExplorer targeting to work as an alerting and reporting tool against cyber-attacks. The proposed system supports features, such as ordering and filtering and vulnerability severity ratings. | The proposed solution provides a solution for secure application development but it can be used after the application development | Y | Y | P | N | 2.5 |
| 13 | Holik et al. [37] | This article explains penetration testing tools and techniques and explains Metasploit framework use in production environment. | The android app challenges and issues were not discussed. | Y | Y | P | N | 2.5 |
| 14 | Cuixiong et al. [38] | The paper presents test automation approach for Android app. testing process, with emphasis on GUI bugs. log/trace files for detection of GUI bugs. | Pentest frame works were not discussed how ever the proposed solution effectively helps in bug's discovery. | P | Y | P | P | 2.5 |
| 15 | Y et al. [41] | The paper explains different penetration test tools, techniques and frameworks and how to perform the penetration testing using different criteria. | The paper covers all the aspects of pen-test frameworks and tools but lack details on how vulnerability modeling helps in secure application development. | Y | Y | p | p | 3 |
| 16 | Karygiannis et al. [43] | A prototype security testing tool of NIST was explained and how it helps in security and penetration testing. | The paper discussed the NIST framework but the Challenges faces by the developer were not discussed. | P | Y | N | N | 1.5 |
| 17 | Navdeep et al. [44] | This paper discussed how security can be integrated in early stages of software development lifecycle and how to model different vulnerabilities. | The paper discussed various aspects of penetration test framework and tools and how modeling helps. | Y | P | P | P | 2.5 |
| 18 | Jiun-kai et al. [45] | An automated Pen-testing tool is proposed and how it can help in detecting vulnerabilities in different platforms. | The paper only covers DVD/USB platforms does not cover the android related vulnerabilities. | Y | Y | P | N | 2.5 |
| 19 | Hung. et al. [49] | The author conducted a survey of various pen-test tools and frameworks. The paper discussed different pen-testing levels like system testing, interface testing and transportation testing. | Vulnerabilities of applications and services are effectively discovered using proposed penetration test framework. The paper does not cover how modeling tools can help securing application. | Y | Y | P | N | 2.5 |
| 20 | Malek et al. [50] | Overview of techniques for increasing the security and reliability of application in an Android applications. A number of test case for fuzzing of applications. | It described analysis techniques from android and how automated test coverage can be done for fuzzing. The paper however, doesn't not discuss how security can be improved during coding stage. | Y | Y | N | P | 2.5 |
| 21 | Martinelli et al. [52] | This paper presents a framework using static and dynamic approaches for precise detection of Android malware. N-gram analysis technique is used for static analysis while multi-monitoring is done for dynamic analysis. | 2794 malicious apps has been tested using the framework and accuracy of 99.7% is reported. The paper however, does not discuss the vulnerability modeling techniques and how model-based tested can be done during intimal stages of SDLC. | Y | Y | Y | N | 3 |
| 22 | Mustapha et al. [57] | Penetration exploits and other vulnerabilities have been discussed. Different tools and techniques have been reviewed for pen-testing. | Pen-test frameworks, tools and techniques have been discussed to enhance the security. The demerits have not been discussed. | Y | Y | P | N | 2.5 |

**TABLE 4.** *(Continued.)* **Literature quality assessment.**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 23 | Rodrguez-mota et al. [59] | The paper proposes two hybrid malware detection frameworks using static-dynamic analysis approach. | Android Hybrid malware detection test framework were discussed. Android developer challenges and their solutions were not discussed. | Y | p | N | N | 1.5 |
| 24 | Sandhya et al. [60] | This paper discussed the need of penetration testing and how Wireshark helps in detecting and testing. The paper also surveyed different penetration tools and techniques. | Adversaries were detected using Wireshark tool and comparison of different frameworks have been provided. The paper does not cover developer challenges. | Y | P | Y | N | 2.5 |
| 25 | Severin et al. [61] | Assessments of Pentation testing tools was done to discover automated tools which can help in vulnerabilities assessment and can cover security flaws. Systematic test approach was specified based on autonomous testing tools. | Different penetration test tools and methodologies along with test coverage were discussed. the paper has not covered the issues and challenges faced by the android developers. | Y | P | P | N | 2 |
| 26 | Shanley et al. [64] | Different penetration testing tools, methodologies and frameworks have been discussed along with comparisons of ISSAF and OWASP. | Domain coverage and issues during the android application development were partially discussed. | Y | Y | P | P | 3 |
| 27 | Shaukat et al. [65] | The Papers provides a survey of different frameworks used in penetration testing and proposed a test technique to secure android applications. | Infiltration testing approach along with survey of framework was discussed and how developers can improve security. | Y | P | P | P | 2.5 |
| 28 | Faily et al. [66] | A survey on challenges and issues faced by the developers have been discussed. | The author discussed developer issues and challenges but the paper lacks discussion on the tools and techniques for pen-testing. | P | P | P | Y | 2.5 |
| *29* | *Avinash et al. [68]* | Vulnerability scanning approach is proposed which provides better coverage in less time with no false positives. | Pentest frame works were not discussed. However, the proposed solution effectively helps in vulnerability discovery. | *P* | *P* | *P* | *P* | *2* |
| *30* | Trajce et. al. [72] | Two pen-test methodologies have been discussed and how social engineering can help attackers to gain access to system and steal user's data. | The paper discussed various aspects of penetration test framework and tools w.r.t. social engineering. | *P* | *P* | *P* | *N* | *1.5* |
| *31* | Andrey et al. [4] | Intermodal vulnerabilities have been discussed along with a new approach for vulnerability detection using dynamic analysis by incorporating intermodal. | The paper does not cover the penetration test frameworks. | *Y* | *Y* | *P* | *N* | *2.5* |
| *32* | *Santhi et al. [78]* | Different pen-test tools of kali Linux were investigated along with comparison of different tools. | The author discussed and demonstrated how vulnerabilities are exploited to gain access to the system. | *Y* | *Y* | *P* | *P* | *3* |
| *33* | *Walden et al. [79]* | The author explained how vulnerabilities effect the system and what guidelines developer should follow for penetration secure development. | Different vulnerabilities along with practical demonstration was discussed. | *P* | *P* | *P* | *P* | *2* |
| *34* | *Wen et al. [80]* | The author discussed various issues which developer encounter. The paper focused on testing efficiency and proposes a GUI testing platform. | No discussion of existing pen-test frameworks and how model base development helps for secure development. | *N* | *Y* | *P* | *P* | *2* |
| *35* | *Xiong et al. [82]* | The authors briefly discussed penetration test frameworks and proposed a model – based framework for penetration testing of application. | No discussion on developer issues. The framework only help in web application vulnerability discovery. | *Y* | *Y* | *P* | *N* | *2.5* |
| *36* | *Vats et al. [83]* | The authors discussed tools techniques for effective penetration testing. | Existing frameworks and developer issues are not discussed. | *Y* | *Y* | *P* | *P* | *3* |
| *37* | Borja et al. [84] | Different pen-test tools were used to perform testing on several android applications in accordance with OWASP mobile common risk. Risk metrics were used to resume main attacker points and vulnerability of mobile devices. | Issues developers face during application development are not discussed. | *Y* | *Y* | *P* | *N* | *2.5* |

**TABLE 4.** *(Continued.)* **Literature quality assessment.**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *38* | *Dawson et al. [23]* | The paper proposed Penetration test methodology using existing approaches and also explained how threat modeling can be used to mitigate threats. | The paper explained various pen-test issues and how developer can secure the application. | *P* | *N* | *Y* | *Y* | *2.5* |
| *39* | *Qu et al. [85]* | The author proposed Dydriod which uses both static and dynamic analysis to investigate dynamic code loading and malicious behavior. | Existing pen-test frameworks were not discussed. | *Y* | *P* | *N* | *N* | *1.5* |
| *40* | *Yerima et al. [86]* | The paper discussed various techniques for increasing code coverage and proposed based on state based method for accurate machine learning based malware detection. | Model-based techniques for secure app development partialy discussed. | *N* | *Y* | *P* | *P* | *2* |
| *41* | *Bojjagani et al. [87]* | The paper discussed how android banking apps can be secured. Different tools are discussed and how they can be used by the developers at each stage of app development are discussed. | The paper moderately discussed all the areas of addroid app development. | *P* | *P* | *P* | *P* | *2* |
| *42* | *Alanda et al. [88]* | The paper discussed how different OWASP techniques can be employed to detect vulnerabilities. | No discussion of existing pen-test frameworks and how model base development helps for secure development. | *Y* | *N* | *N* | *P* | *1.5* |
| *43* | *Aymaz et al. [9]* | The author discussed different load balancing issues in servers and implement new strategy and analyzed it using wireshark. | Model based testing and developer issues were not discussed. | *Y* | *Y* | *P* | *N* | *2.5* |
| *44* | *Shi et al. [90]* | The paper briefly discussed pen-test tools, methodologies and its short comings. Combined with cyber security and fingerprinting technology a new framework is introduced. | The paper covers most of the issues related to pen-testing and secure development. | *Y* | *Y* | *P* | *P* | *3* |
| *45* | *Mburano et al. [91]* | Different vulnerability scanners were tested and evaluated using OWASP benchmark. | Developer challenges faced during application development are partially discussed. | *Y* | *Y* | *N* | *P* | *2.5* |
| *46* | *Seyyar et al. [92]* | Using server logs, the author proposed a technique to detect attack-oriented scans on the real-time log data of Apache web server. | Pen-test frame works are not discussed how ever the proposed solution effectively helps in attack discovery. | *P* | *N* | *Y* | *P* | *2* |
| *47* | *Aslan et al. [93]* | Different malware detection and pen-test tools are evaluated and based on results an affective methodology is proposed based on static and dynamic analysis. | Existing frameworks and developer issues are not discussed. | *Y* | *P* | *N* | *N* | *1.5* |
| *48* | *Jain et al. [94]* | Framework comprising of vulnerability discovery and vulnerability scanning approach is proposed. | Pen-test frameworks and developer challenges are not discussed. | *Y* | *P* | *P* | *N* | *2* |
| *49* | *Yang et al. [45]* | The author proposed an automated tool to discover vulnerabilities and weaknesses of the system suing Live DVD/USB platform. | The android app challenges and issues were not discussed. | *Y* | *P* | *N* | *N* | *1.5* |
| 50 | *Halfond et al.* [99] | The paper proposed Static and Dynamic Analysis based penetration testing approach using input vector to detect vulnerabilities. | The paper has not explained how vulnerability modeling helps. | Y | P | N | P | 2 |
| 51 | *Clapp et al.* [100] | Automatic test techniques are explores and a delta debugging variant is proposed to sale the trace minimization problem. | Framework + tools are not discussed by the author. | P | N | N | P | 1.5 |
| 52 | *Pakevicius et al.* [101] | The paper presented an automated for user interface defects detection using a list of predefined UI defects. | The paper explained various pen-test issues and how UI of android app can be secured. | P | P | P | p | 2 |
| 53 | *Palma et al.* [102] | An awareness based system is proposed to help developers to improve security and quality of android applications. | Pen-test frame works were not discussed. | P | N | P | P | 1.5 |

**TABLE 4.** *(Continued.)* **Literature quality assessment.**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 54 | *Mahmood et al.* [103] | For detection of vulnerabilities, EvoDroid is proposed for program and algorithm analysis of android apps with increased code coverage. | The paper lacks discussion on the tools and techniques for pen-testing. | Y | Y | P | N | 2.5 |
| 55 | *Asaad et al.* [104] | The study identified weaknesses by penetrating in the system using different attacks scenarios. | Android attack scenarios were not discussed. | Y | P | P | N | 2 |
| 56 | *Vats et al.* [83] | The paper review different penetration test tools, techniques and frameworks in terms of compatibility, usability and utility for effective vulnerability identification. | The paper explained various pen-test tools and issues for identifying vulnerabilities and securing the application. | Y | Y | Y | p | 3.5 |

penetration testing point of view. Whereas, OWASP covers all the aspects of web, software and mobile application development domains and can be implemented in all the phases of SDLC. NIST allows profiling for implementing flexibility in the framework by defining cybersecurity outcomes to achieve the desired risk management goals. OSSTMM, OWASP, ISSAF and PTES have limited flexibility in various pen-test execution criteria.

OWASP and OSSTMM have well defined and detailed processes and guidelines for covering differing vulnerabilities in software applications. OWASP framework allows and provides guidelines that can help the developers to implement security in all the phases of SDLC.

NIST, OSSTMM, OWASP, ISSAF and PTES do not mention any modelling tools nor do they provide any technique on how to embed security and design different threat models during designing of the application. OWASP and NIST provide a guideline on vulnerabilities and how to mitigate certain risks by secure coding but none of the above has the knowledge base and tools which can help the designers and developer to design a secure application by automatically identifying vulnerabilities during the modelling of an application.

Based on the evaluation tabulated in Table 5, it is evident that these penetration test frameworks provide guidelines, tools and techniques to assess security and focus on penetration testing after the completion of an application. These methodologies are generic and mainly focused on web-based applications. Developers and testers cannot use them without having the expertise and knowledge of change effect. Hence, it is believed no such model-driven methodology for penetration testing in an Android application is available which can help the designer to use a threat-model knowledge-base to model vulnerabilities.

## 5. Penetration test Tools (RQ1-part 2)

Penetration testing tools help in identifying and eliminating security issues present in the application. The model-based penetration test methodology is performed by auto-generated test cases on the basis of SUT models. During our systematic literature review, it was revealed that more than 50% of the tools used model-based methodology and test cases were automatically generated and executed. We observed

**TABLE 5.** **Pen-test methodologies and frameworks.**

| | | Pen-Test Methodologies and frameworks | | | | |
|---|---|---|---|---|---|---|
| | | **OSS TMM 3** | **NIST-ISAM** | **OWASP-MASVS** | **ISSAF** | **PTES** |
| ISO/IEC 25010:2013 software quality model | **Coverage** | A | PA | A | PA | PA |
| | **Flexibility** | PA | A | PA | PA | PA |
| | **Adoptability** | PA | PA | A | NA | PA |
| | **Modelling** | NA | NA | NA | NA | NA |
| | **Usability for amateur developers** | NA | PA | PA | NA | PA |

Legend : A: Applicable P: Partially Applicable NA: Not Applicable

during the review that these pen-test tools use various test methodologies which include model-based, search-based, fuzzing and mutation-based testing. For better test coverage, there are many tools which use multiple test methodologies (EVODRIOD) [111], [112].

There are three types of testing 1) White Box testing is done at the early stages by testing internal structures. Full knowledge of source code is required to develop test cases. 2) Black box testing tests the functionality of the application and 3) Gray box testing can be done with limited knowledge of internal structure and functionality. i.e. for android apps, tests can be developed by going through manifest file. Vega at el. [112] benchmarked tools by dynamic analysis of vulnerabilities and observer that knowledge bases and search algorithm are periodically updated for improvement in accuracy but they still leave some vulnerabilities unaccounted. Although reviewers and bloggers publish the review about tools accuracy in various websites, it is difficult for the developers and users to know which tool is suitable for them and what sort of vulnerabilities are captured. Munoz *et al.* [113] have compared the accuracy of model-based penetration test tools and compared the result of the vulnerability reported by the tools. It was observed that a number of tools are not accurate and show false-positive results.

During our research, we identified and analyzed following thirteen (13) tools from the selected literature.

- **Metasploit [89]** is a GUI and command-line tool and has an advance framework for penetration testing. It uses a payload to perform exploit on target machines and can be used on network application and servers etc. Metasploit can also perform manual brute-forcing and static and dynamic code analysis.
- **Core Impact** is used for mobile device penetration testing. It also performs network device penetration through password identification and cracking [45].
- **W3af** is a web penetration framework that helps in penetration testing through vulnerability discovery, attack and vulnerability audit. W3af has proxy support and performs HTTP response cache, DNS cache, and cookie handling and user agent faking [94].
- **Wireshark** helps in network analysis and it can capture and display packets in real-time. Packets can also be captured offline and network packet analyzer provides details about packet, protocols etc. Wireshark is multiplatform and runs on various operating systems and provides decryption support for SSL, TLS, WPA and IPSec [89].
- **Netsparker** has a web scanner that helps in identification of vulnerabilities and provides a solution. It also detects and exploits SQL injections and local file inductions [92].
- **BurpSuite** performs sensitive data searching and IP scans. It also performs web scanning to detect vulnerabilities. Burp Suite is multiplatform and also intercepts crawling contents [93].
- **OWASP ZAP** identifies security holes through passive scanning and also perform brute force attack for password cracking and directories access. ZAP identifies and exploits vulnerabilities and can execute Beanshell scripts [91].
- **Nmap** is a multi-platform security scanner to detect network host and displays results in the form of interactive graphs. It can draw topologies and shows changes in the network hosts and services to track any vulnerabilities or services which are down [88].
- **Intent Fuzzer** is android vulnerability scanner and finds the bug that can crash the system. It fuzzes all the components including broadcast receivers. It can only fuzz a single activity at a time [102].
- **Caiipa** runs the app in real-time hardware or in the emulator and performs bugs and fuzzing test. Another important aspect of Caiipa is that it inspects the code as well to check if it is draining the system resources [100].
- **IntentDroid** [107] dynamically analyses Android apps for Inter App Communication related vulnerabilities. It generates attack scenario for vulnerabilities such as User-Interface, SQL Injection, Unsafe Reflections, Spoofing and Cross-Site Scripting.
- **CRAXDroid** tests Android mobile and tablets and test system and applications in executable form without source code and uses Android emulator. CraxDroid automatically crawls execution paths to identify potential software defects. [101].
- **Sapienz** performs White, Gray and Black box testing on the application under test. It also unpacks and repacks the app and uses non-invasive skin coverage [100].

### A. EVALUATION CRITERIA

As shown in table 6, the above tools were evaluated using the below-mentioned criteria.

**Code Analysis:** Reconnaissance – Information gathering on the target and footprinting.

**Code Review:** Identifying issues in source code.

**Vulnerability Analysis:** Capability of identifying, classifying and prioritizing vulnerabilities.

**Vulnerability Exploit:** Proficiency of exploiting the vulnerability.

**Payload:** Reports on test results.

**Model bases test generation:** The tool automatically generates test cases.

**Vulnerability modelling in the design phase using UML:** The tools supports the modeling of threat vectors.

If the tool supports (✓) the assessment criteria the tool will get 1 score and if not supported (✕ ) will get 0 score. Since we have seven evaluations criteria so the total score is 7.

### B. EVALUATION RESULTS

Analysis of penetration test tools mentioned in Figure 6 revealed that these tools can detect vulnerabilities from applications after development of application. However, any vulnerabilities detected can be fixed by the developer post application development which cost extra time and often lead to changes in the entire structure. Among selected tools, we were unable to find functionality which can help in producing a secure design by integrating security stereotypes during the design phase of android application for novice developers/designers who have little or no knowledge about a secure design or familiarity around penetration related issues.

## 6. Conclusion and Discussion

Different penetration test frameworks and tools were analyzed and each framework has different coverage issues for Android domain. Although, these frameworks provide guidelines, tools and techniques for effective penetration testing but these methodologies are generic and mainly focused on web-based applications. Developers and testers cannot use them without having the expertise and knowledge of change effect. For effective penetration testing and to test vulnerabilities in the application, all the policy base, static and dynamic approaches and techniques should be applied to overcome limitations in penetration testing tools. Different communication channels such as shared preferences and Content provider should also be considered for analysis. Below are a few findings for effectively testing the application for vulnerabilities.

- The security level must be decided at the beginning of the application development and Secure development techniques should be employed.

**TABLE 6.** Comparison of tools.

| | Purpose | Code Analysis | Code Review | Vulnerability Analysis | Vulnerability Exploit | Payload | Model-Based | Vulnerability Modeling in Design Phase | Total Score |
|---|---|---|---|---|---|---|---|---|---|
| **Metasploit [90]** | GUI and command-line tool , Performs Payload exploit on Local and network target machine, Static and Dynamic code analysis | ✓ | ✓ | ✓ | ✓ | ✓ | × | × | 5 |
| **Core Impact [45]** | Mobile device penetration testing, network device penetration through password identification and cracking | ✓ | × | ✓ | ✓ | ✓ | × | × | 4 |
| **w3af [94]** | Web penetration framework, Performs vulnerability discovery, attack and vulnerability audit, performs HTTP response cache, DNS cache, and cookie handling and user agent faking | × | × | ✓ | ✓ | ✓ | × | × | 3 |
| **Wireshark [89]** | Network analysis in real-time, multiplatform , provides decryption support for SSL, TLS, WPA and IPSec | × | × | ✓ | ✓ | ✓ | × | × | 3 |
| **Netsparker [92]** | Web scanner, helps in identification of vulnerabilities and provides a solution, detects and exploits SQL injections and local file inductions | × | × | ✓ | × | ✓ | × | × | 2 |
| **Burp Suite [93]** | Web scanning, Performs sensitive data searching and IP scans. , multiplatform and also intercepts crawling contents | ✓ | × | ✓ | ✓ | ✓ | × | × | 4 |
| **OWASP ZAP [91]** | Identifies security holes through passive scanning , perform brute force attack for password cracking and directories access, identifies and exploits vulnerabilities | ✓ | ✓ | ✓ | ✓ | ✓ | × | × | 5 |
| **Nmap [88]** | Multi-platform, detects vulnerabilities on network host, tracks any vulnerabilities or services which are down | × | × | ✓ | ✓ | ✓ | × | × | 3 |
| **SAPIENZ [100]** | White, Gray and Black box testing, unpacks and repacks the app and uses non-invasive skin coverage | × | × | ✓ | ✓ | ✓ | ✓ | × | 4 |
| **IntentDroid [101]** | Dynamically examines Android apps, generates attack scenario for vulnerabilities, analyzer for Activity component of apps | × | × | ✓ | ✓ | ✓ | ✓ | × | 4 |
| **IntentFuzzer [102]** | Android vulnerability scanner, finds the bug that can crash the system. fuzzes all the components including broadcast receivers. | ✓ | × | ✓ | ✓ | ✓ | × | × | 4 |
| **Caiipa [103]** | Runs the app in real-time hardware or in the emulator and performs bugs and fuzzing test, inspects the code to check if it is draining the system resources | ✓ | × | ✓ | ✓ | ✓ | ✓ | × | 4 |
| **CRAXDroid [107]** | Identifies vulnerabilities in android based phones and tablets, automatically crawls execution paths to identify potential software defects | × | × | ✓ | ✓ | ✓ | ✓ | × | 4 |

Legend : ✓: Supported ×: Not Supported

- Sensitive data and information should be identified and handled as per security requirements.
- Functional tests should be performed based on the system requirements.
- Both Manual and automated testing tools should be used to test security aspects of the android application.
- Test bed of the applications should not only be limited to the specific application but it should also cover IPC (inter process communication) and privilege handling among different applications.
- Automated code analysis and code review techniques should be employed.
- Different methods like GET, POST, PUT etc. along with functionality, data format and data shared between services should be tested.
- Application should be tested for memory and energy leaks.

Based on the research and literature review, it is concluded that no such model-driven methodology for penetration testing in an Android application is available which can help the developer to model vulnerabilities during the design and identify risks during coding stage. There is an urgent need to develop solutions which can monitor the source code of the app in real time and can capture and prompt coding vulnerabilities during the development stage.

The researcher community can focus on developing an application which can have inbuilt threat repository to help developer in designing the functionality using custom secure profile of UML stereotypes for identifying all the threats/vulnerabilities during coding as well as code auto correct feature using Java libraries.

## V. APP DEVELOPMENT CHALLANGES (RQ2)

Mobile application development involves writing software applications that can be native or web and uses the same SDLC. Android devices are different based on hardware features and developer must ensure that app should be capable to run on different devices carrying the same android OS version. As shown in Figure 7, our literature review concluded that android app developers come across a number of challenges w.r.t android security and privacy.

### A. ADHERENCE TO BEST PRACTICES

Malicious applications in android OS can use other applications to steal data and drain resources [114]. Various android security frameworks provide best practices and vulnerability

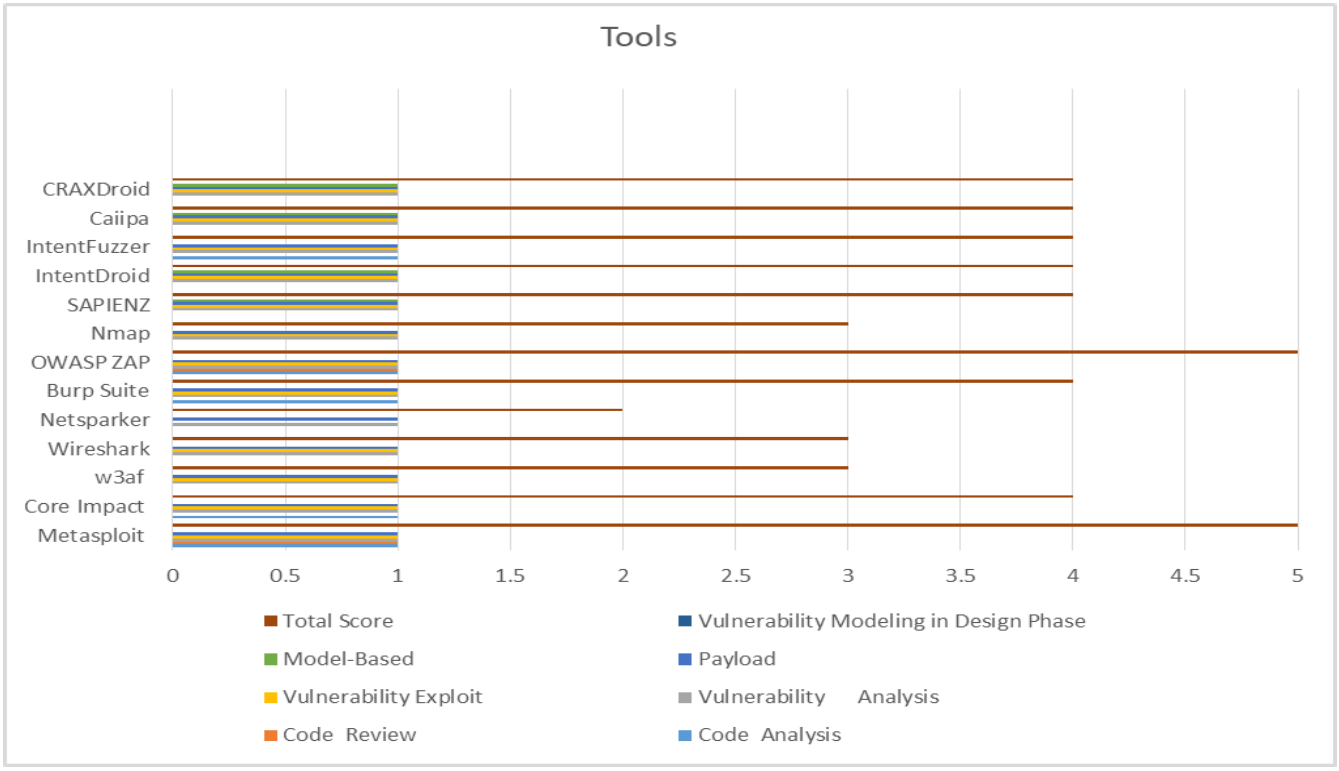



**FIGURE 6.** Evaluation results of testing tools.

documentation but developers do not adhere to these best practices and resulting applications are often vulnerable to attacks. Android apps are built using third party exposed components and devices. The developer will be responsible for data security and privacy leak if these components are not protected through label declaration [115], [116].

### B. LIMITING COMPUTING POWER AND RESOURCES

In mobile app design, a major hindrance is the screen size, limited processing power and limited storage space. A set of best practices guidelines to deal with issue and challenges related to mobile app development can greatly help the designer and developer. For application where developers do not take proper care of mobile resource utilization, battery and resource drain may occur [117]. Design limitation often results in development problems [118].

### C. LACK OF SECURITY KNOWLEDGE AND COMPLEXITY OF TESTING

Due to limited resources and the unique nature of devices, the mobile app testing faces a number of issues. Applications often tested with mobile emulator often do not generate the same real-time device characteristics and do not show network and hardware related failure due to coding issues. There is a need for a model-driven testing framework for Automated testing functional and compatibility testing of mobile phone apps [119].

### D. SECURITY AND PRIVACY

Since the era has changed, Web applications are mostly converted into mobile apps. Android apps are widespread and it's the developer's responsibility to protect and secure user data by inducing end to end security during the mobile app development. These applications should be tested thoroughly for security through automated testing tools and test cases [120]. Lack of security testing tools and techniques also affect the reliability of resulting android application [122]. Security flaws emerge from time to time in various android versions such as Fakeid, mRST and Hijacking etc. developers often do not have knowledge of such security flaws which result in a vulnerable application.

### E. MANIFEST FILE CONFIGURATIONS ISSUES

The Manifest file is written by developers and contains important parameters and configurations to run an application. These configurations include security, privacy and accessibility of an application. Any type of permissions which are required to access protected parts of system are defined in the manifest and can lead to serious privacy and security infringes if not handled properly by developer [121].

*Discussion:*

Mobile application development is a new field and a number of programmers are shifting to mobile app development. Security requirements are not captured during development and penetration testing since it is considered as nonfunctional requirement and is performed at the time of deployment.

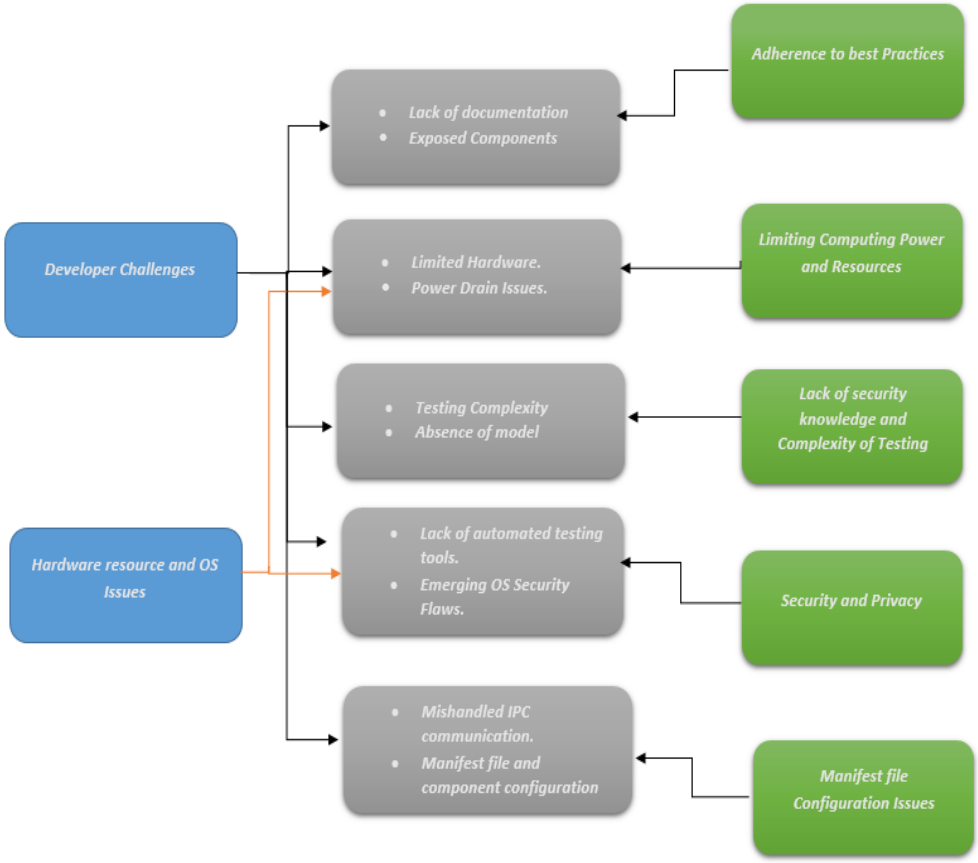



**FIGURE 7.** Android developer challenges.

This makes the application vulnerable and late pen-testing and test and patch technique also consumes more time and cost.

The research indicates that developers lack security awareness and make ad-hoc decisions for implementation of security during the development stage. Data encryption and other recommended security and privacy measures and best-practices are not adhered. Third party tools are often used for penetration testing which do often cover the complete app environment as well as escalated provisions and Inter process communications. Non authentic libraries often used which comprise privacy and result in loss of user's data. Security is not considered during the early phases of software development life cycle. Penetration testing is considered as a non-functional activity and lack of pen-tests result in a faulty application. Unified modeling language doesn't allow to model android attack vector as it does not have specific elements to model these interaction for android environment.

Based on the research it is evident that developers can adopt following practices to secure the application.

- Write efficient and secure code using best coding practices.
- Data should be encrypted and inter process communication should be secured.
- Code of third party libraries should be thoroughly tested for vulnerabilities before embedding them.
- High level authentication code and policies should be implemented.
- Security policies should be implemented at each stage of SDLC.
- Application should have a mechanism to detect code tempering.
- Use of modern malware detetion tools with deep leanrning abiliteis can secure the application.
- Minimum privileges should be assigned to the application and sessions should be handled properly.

## VI. FUTURE DIRECTION

The fast-paced android app market is facilitating the businesses but at the same time it is alarming that a number of applications are vulnerable. In this research, we have discussed many frameworks and tools for android penetrations testing. We have also discussed various developers issues related to android application development. Based on the research and answers to the research questions, we will make the following contributions to help the developers to produce time-efficient and secure applications:

- Highly proficient hybrid learning enabled intelligent multivector malware detection mechanism.

- A novel model-based Android Pernetration secure framework which can out perform both in terms of time efficiency and detection accuracy.
- A vulnerability repository which can be updated to provide tooltips while designing the application using Java classes to identify related threats.

## REFERENCES


[1] M. Howard, "The security development lifecycle," Redmond, WA, USA: Microsoft Press, 2006.

[2] A. Al-Ghamdi, "A survey on software security testing techniques," *Int. J. Comput. Sci. Telecommun.*, vol. 4, pp. 14–18, Apr. 2013.

[3] 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)*, Farmingdale, NY, USA, Apr. 2016, pp. 1–6, doi: 10.1109/LISAT.2016.7494156.

[4] A. Petukhov and D. Kozlov, "Detecting security vulnerabilities in Web applications using dynamic analysis with penetration testing," in *Proc. Appl. Secur. Conf.*, 2008.

[5] B. Arkin, S. Stender, and G. McGraw, "Software penetration testing," *IEEE Secur. Privacy Mag.*, vol. 3, no. 1, pp. 84–87, Jan. 2005, doi: 10.1109/msp.2005.23.

[6] Y. Arya, A. Bhalotiya, C. Sharma, V. Kag, and S. Snaghvi, "International journal of engineering sciences & research technology a study of metasploit tool," *Arya*, vol. 5, p. 2, Feb. 2016.

[7] A. G. Bacudio, X. Yuan, B. T. Bill Chu, and M. Jones, "An overview of penetration testing," *Int. J. Netw. Secur. Its Appl.*, vol. 3, no. 6, pp. 19–38, Nov. 2011, doi: 10.5121/ijnsa.2011.3602.

[8] P. Baker and C. Jervis, "Early UML model testing using TTCN-3 and the UML testing profile," in *Proc. Test., Acad. Ind. Conf. Pract. Res. Techn. (TAICPART-MUTATION)*, Sep. 2007, pp. 47–54.

[9] B. Baloglu, "How to find and fix software vulnerabilities with coverity static analysis," *IEEE Cybersecurity Develop. (SecDev)*, Nov. 2016, p. 153.

[10] A. Bartel, J. Klein, Y. L. Traon, and M. Monperrus, "Automatically securing permission-based software by reducing the attack surface: An application to Android," in *Proc. 27th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2012, pp. 274–277.

[11] D. D. Bertoglio and A. F. Zorzo, "Overview and open issues on penetration test," *J. Brazilian Comput. Soc.*, vol. 23, no. 1, pp. 1–16, Dec. 2017.

[12] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015.

[13] J. Bo, L. Xiang, and G. Xiaopeng, "MobileTest: A tool supporting automatic black box test for software on smart mobile devices," in *Proc. 2nd Int. Workshop Automat. Softw. Test*, May 2007, p. 8.

[14] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Softw. Eng. Group, Tech. Rep., 2007.

[15] J. Burns, "Developing secure mobile applications for Android," ISEC, Tech. Rep., 2008.

[16] C. Carthern, W. Wilson, R. Bedwell, and N. Rivera, "Introduction to network penetration testing," in *Cisco Networks*. Berkeley, CA, USA: Apress, pp. 759–772, 2015.

[17] Y. Cheon, "Are java programming best practices also best practices for Android," Dept. Comput. Sci., Univ. Texas El Paso, El Paso, TX, USA, Tech. Rep. 16-76, 2016.

[18] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner, "Analyzing inter-application communication in Android," in *Proc. 9th Int. Conf. Mobile Syst., Appl., services (MobiSys)*, Jun. 2011, pp. 239–252.

[19] H. M. Z. A. Shebli and B. D. Beheshti, "A study on penetration testing process and tools," in *Proc. IEEE Long Island Syst., Appl. Technol. Conf. (LISAT)*, Farmingdale, NY, USA, May 2018, pp. 1–7, doi: 10.1109/LISAT.2018.8378035.

[20] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 4, pp. 586–602, Oct. 2017, doi: 10.1109/TETC.2016.2606384.

[21] G. W. Cross, "Using a protocol analyzer to introduce communications protocols," in *Proc. 10th ACM Conf. SIG-Inf. Technol. Educ. (SIGITE)*, Oct. 2009, pp. 178–181.

[22] D. Menoski, P. Mitrevski, and T. Dimovski, "Evaluating Website security with penetration testing methodology," in *Proc. Int. Conf. Appl. Internet Inf. Technol.*, Zrenjanin, Serbia, 2014.

[23] J. Dawson and J. T. McDonald, "Improving penetration testing methodologies for security-based risk assessment," in *Proc. Cybersecurity Symp. (CYBERSEC)*, Apr. 2016, pp. 51–58.

[24] R. Shanmugapriya, "A study of network security using penetration testing," in *Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES)*, Feb. 2013, pp. 371–374, doi: 10.1109/ICICES.2013.6508375.

[25] B. Duan, Y. Zhang, and D. Gu, "An easy-to-deploy penetration testing platform," in *Proc. 9th Int. Conf. Young Comput. Scientists*, Nov. 2008, pp. 2314–2318.

[26] X. Qiu, S. Wang, Q. Jia, C. Xia, and Q. Xia, "An automated method of penetration testing," in *Proc. IEEE Comput., Commun. IT Appl. Conf.*, Oct. 2014, pp. 211–216.

[27] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan, "Android security: A survey of issues, malware penetration, and defenses," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 998–1022, 2nd Quart., 2015, doi: 10.1109/comst.2014.2386139.

[28] N. Gandhewar and R. Sheikh, "Google Android: An emerging software platform for mobile devices," *Int. J. Comput. Sci. Eng.*, vol. 1, no. 1, pp. 12–17, 2010.

[29] (2019). *In Gartner*. Accessed: Jul. 11, 2019. [Online]. Available: https://www.gartner.com/reviews/market/mobile-application-security-testing

[30] S. Gejibo, F. Mancini, K. A. Mughal, R. A. B. Valvik, and J. Klungsøyr, "Secure data storage for mobile data collection systems," in *Proc. Int. Conf. Manage. Emergent Digit. EcoSystems (MEDES)*, 2012, pp. 131–144.

[31] R. Goel, M. C. Govil, and G. Singh, "A secure software design methodology," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2016, pp. 2484–2488.

[32] (2019). *In: Omg.Org*. Accessed: Sep. 19, 2019. [Online]. Available: https://www.omg.org/spec/UML/2.3/Infrastructure/PDF

[33] H. Gupta and R. Kumar, "Protection against penetration attacks using metasploit," in *Proc. 4th Int. Conf. Rel., Infocom Technol. Optim. (ICRITO) (Trends Future Directions)*, Sep. 2015, pp. 1–4.

[34] M. Gusev, S. Ristov, and A. Donevski, "Security vulnerabilities from inside and outside the eucalyptus cloud," in *Proc. 6th Balkan Conf. Informat. (BCI)*, Sep. 2013, pp. 95–101.

[35] L. Harrison, R. Spahn, M. Iannacone, E. Downing, and J. R. Goodall, "NV: Nessus vulnerability visualization for the Web," in *Proc. 9th Int. Symp. Visualizat. Cyber Secur. (VizSec)*, Oct. 2012, pp. 25–32.

[36] V. Y. Hnatyshin and A. F. Lobo, "Undergraduate data communications and networking projects using opnet and wireshark software," *ACM SIGCSE Bull.*, vol. 40, no. 1, p. 241, 2008, doi: 10.1145/1352322.1352222.

[37] F. Holik, J. Horalek, O. Marik, S. Neradova, and S. Zitta, "Effective penetration testing with metasploit framework and methodologies," in *Proc. IEEE 15th Int. Symp. Comput. Intell. Informat. (CINTI)*, Nov. 2014, pp. 237–242.

[38] C. Hu and I. Neamtiu, "Automating GUI testing for Android applications," in *Proc. 6th Int. Workshop Automat. Softw. Test*, May 2011, pp. 77–83.

[39] L. F. Johnson and H. Witchey, "The 2010 horizon report: Museum edition," *Curator, Museum J.*, vol. 54, no. 1, pp. 37–40, Jan. 2011, doi: 10.1111/j.2151-6952.2010.00064.x.

[40] G. K. Juneja, "Ethical hacking: A technique to enhance information security," *Int. J. Innov. Res. Sci., Eng. Technol.*, vol. 2, no. 12, pp. 7575–7580, 2013.

[41] Y. Kang, H. H. Cho, Y. Shin, and J. B. Kim, "Comparative study of penetration test methods," *Adv. Sci. Technol. Lett.*, vol. 87, no. 1, pp. 34–37, 2015.

[42] R. Kapoor and S. Agarwal, "Sustaining superior performance in business ecosystems: Evidence from application software developers in the iOS and Android smartphone ecosystems," *Org. Sci.*, vol. 28, no. 3, pp. 531–551, Jun. 2017, doi: 10.1287/orsc.2017.1122.

[43] T. Karygiannis, "Network security testing using mobile agents," in *Proc. 3rd Int. Conf. Exhib. Practical Appl. Intell. Agents Multi-Agent Technol.*, London, U.K., Mar. 1998, pp. 1–7.

[44] N. Kaur and P. Kaur, "Mitigation of SQL injection attacks using threat modeling," *ACM SIGSOFT Softw. Eng. Notes*, vol. 39, no. 6, pp. 1–6, 2014, doi: 10.1145/2674632.2674638.

[45] J.-K. Ke, C.-H. Yang, and T.-N. Ahn, "Using w3af to achieve automated penetration testing by live DVD/live USB," in *Proc. Int. Conf. Hybrid Inf. Technol. (ICHIT)*, May 2009, pp. 460–464.

[46] M. Ko, Y.-J. Seo, B.-K. Min, S. Kuk, and H. Soo Kim, "Extending UML meta-model for Android application," in *Proc. IEEE/ACIS 11th Int. Conf. Comput. Inf. Sci.*, May 2012, pp. 669–674.

[47] L. Liu, J. Xu, C. Guo, J. Kang, S. Xu, and B. Zhang, "Exposing SQL injection vulnerability through penetration test based on finite state machine," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Oct. 2016, pp. 1171–1175.
[48] N. Mahendra and S. Ahmad, "A categorized review on software security testing," *Int. J. Comput. Appl.*, vol. 154, no. 1, pp. 21–25, Nov. 2016, doi: 10.5120/ijca2016912023.
[49] C.-K. Chen, Z.-K. Zhang, S.-H. Lee, and S. Shieh, "Penetration testing in the IoT age," *Computer*, vol. 51, no. 4, pp. 82–85, Apr. 2018.
[50] S. Malek, N. Esfahani, T. Kacem, R. Mahmood, N. Mirzaei, and A. Stavrou, "A framework for automated security testing of Android applications on the cloud," in *Proc. IEEE 6th Int. Conf. Softw. Secur. Rel. Companion*, Jun. 2012, pp. 35–36.
[51] Y. V. N. Manikanta and A. Sardana, "Protecting Web applications from SQL injection attacks by using framework and database firewall," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, 2012, pp. 609–613.
[52] F. Martinelli, F. Mercaldo, and A. Saracino, "BRIDEMAID: An hybrid tool for accurate detection of Android malware," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 899–901.
[53] M. Felderer, P. Zech, R. Breu, M. Büchler, and A. Pretschner, "Model-based security testing: A taxonomy and systematic classification," *Softw. Test., Verification Rel.*, vol. 26, no. 2, pp. 119–148, Mar. 2016.
[54] M. Mirjalili, A. Nowroozi, and M. Alidoosti, "A survey on Web penetration test," *Adv. Comput. Sci., Int. J.*, vol. 3, no. 6, pp. 107–121, 2014.
[55] O. Olsen, P. Middleton, J. Ezzo, P. C. Gotzsche, V. Hadhazy, A. Herxheimer, J. Kleijnen, and H. McIntosh, "Quality of cochrane reviews: Assessment of sample from 1998," *BMJ*, vol. 323, no. 7317, pp. 829–832, Oct. 2001, doi: 10.1136/bmj.323.7317.829.
[56] A. D. Oxman, "Systematic reviews: Checklists for review articles," *BMJ*, vol. 309, no. 6955, pp. 648–651, Sep. 1994, doi: 10.1136/bmj.309.6955.648.
[57] M. Refai, "Exploiting a buffer overflow using metasploit framework," in *Proc. Int. Conf. Privacy, Secur. Trust Bridge Gap Between PST Technol. Bus. Services (PST)*, Oct. 2006, p. 74.
[58] B. Rexha, A. Halili, K. Rrmoku, and D. Imeraj, "Impact of secure programming on Web application vulnerabilities," in *Proc. IEEE Int. Conf. Comput. Graph., Vis. Inf. Secur. (CGVIS)*, Nov. 2015, pp. 61–66.
[59] A. Rodriguez-Mota, P. J. Escamilla-Ambrosio, S. Morales-Ortega, M. Salinas-Rosales, and E. Aguirre-Anaya, "Towards a 2-hybrid Android malware detection test framework," in *Proc. Int. Conf. Electron., Commun. Comput. (CONIELECOMP)*, Feb. 2016, pp. 54–61.
[60] S. Sandhya, S. Purkayastha, E. Joshua, and A. Deep, "Assessment of website security by penetration testing using wireshark," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Jan. 2017, pp. 1–4.
[61] L. Yang, H. Zhang, H. Shen, X. Huang, X. Zhou, G. Rong, and D. Shao, "Quality assessment in systematic literature reviews: A software engineering perspective," *Inf. Softw. Technol.*, vol. 130, Feb. 2021, Art. no. 106397, doi: 10.1016/j.infsof.2020.106397.
[62] S. Leonhardt, J. Petersohn, and C. Schmid, "Penetration testing," STL GmbH Stuttgart, Stuttgart, Germany, Tech. Rep., 2011.
[63] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer, "Google Android: A comprehensive security assessment," *IEEE Secur. Privacy Mag.*, vol. 8, no. 2, pp. 35–44, Mar. 2010, doi: 10.1109/msp.2010.2.
[64] A. Shanley and M. N. Johnstone, "Selection of penetration testing methodologies: A comparison and evaluation," presented at the Austral. Inf. Secur. Manage. Conf., 2015.
[65] K. Shaukat, A. Faisal, R. Masood, A. Usman, and U. Shaukat, "Security quality assurance through penetration testing," in *Proc. 19th Int. Multi-Topic Conf. (INMIC)*, Dec. 2016, pp. 1–6.
[66] S. Faily, J. McAlaney, and C. Iacob, "Ethical dilemmas and dimensions in penetration testing," in *Proc. HAISA*, Jun. 2015, pp. 233–242.
[67] M. B. Shuaibu and R. A. Ibrahim, "Web application development model with security concern in the entire life-cycle," in *Proc. 4th IEEE Int. Conf. Eng. Technol. Appl. Sci. (ICETAS)*, Nov. 2017, pp. 1–6.
[68] A. K. Singh and S. Roy, "A network based vulnerability scanner for detecting SQLI attacks in Web applications," in *Proc. 1st Int. Conf. Recent Adv. Inf. Technol. (RAIT)*, Mar. 2012, pp. 585–590.
[69] Y. Stefinko, A. Piskozub, and R. Banakh, "Manual and automated penetration testing. Benefits and drawbacks. Modern tendency," in *Proc. 13th Int. Conf. Modern Problems Radio Eng., Telecommun. Comput. Sci. (TCSET)*, Feb. 2016, pp. 488–491.
[70] S. Holla and M. M. Katti, "Android based mobile application development and its security," *Int. J. Comput. Trends Technol.*, vol. 3, no. 3, pp. 486–490, 2012.
[71] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro, "The evolution of Android malware and Android analysis techniques," *ACM Comput. Surveys*, vol. 49, no. 4, pp. 1–41, Feb. 2017, doi: 10.1145/3017427.
[72] T. Dimkov, W. Pieters, and P. Hartel, "Two methodologies for physical penetration testing using social engineering," in *Proc. 26th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Dec. 2010, pp. 399–408.
[73] V. Tilemachos and C. Manifavas, "An automated network intrusion process and countermeasures," in *Proc. 19th Panhellenic Conf. Informat.*, Oct. 2015, pp. 156–160.
[74] I. A. Tondel, M. G. Jaatun, and P. H. Meland, "Security requirements for the rest of us: A survey," *IEEE Softw.*, vol. 25, no. 1, pp. 20–27, Jan. 2008, doi: 10.1109/ms.2008.19.
[75] E. Ungan, S. Trudel, and L. Poulin, "Using FSM patterns to size security non-functional requirements with COSMIC," in *Proc. 27th Int. Workshop Softw. Meas. 12th Int. Conf. Softw. Process Product Meas.*, Oct. 2017, pp. 64–76.
[76] M. Usman, M. Z. Iqbal, and M. U. Khan, "A product-line model-driven engineering approach for generating feature-based mobile applications," *J. Syst. Softw.*, vol. 123, pp. 1–32, Jan. 2017, doi: 10.1016/j.jss.2016.09.049.
[77] S. Vaupel, G. Taentzer, R. Gerlach, and M. Guckert, "Model-driven development of mobile applications for Android and iOS supporting role-based app variability," *Softw. Syst. Model.*, vol. 17, no. 1, pp. 35–63, Feb. 2018.
[78] B. L. V. V. Kumar, K. R. Kumar, and V. Santhi, "Penetration testing using linux tools: Attacks and defense strategies," *Int. J. Eng. Res. Technol.*, vol. 5, no. 12, Dec. 2016. [Online]. Available: http://dx.doi.org/10.17577/IJERTV5IS120166
[79] J. Walden, "Integrating Web application security into the IT curriculum," in *Proc. 9th ACM SIGITE Conf. Inf. Technol. Educ. (SIGITE)*, Oct. 2008, pp. 187–192.
[80] H.-L. Wen, C.-H. Lin, T.-H. Hsieh, and C.-Z. Yang, "PATS: A parallel GUI testing framework for Android applications," in *Proc. IEEE 39th Annu. Comput. Softw. Appl. Conf.*, vol. 2, Jul. 2015, pp. 210–215.
[81] E. Woods, "Harnessing UML for architectural description—The context view," *IEEE Softw.*, vol. 31, no. 6, pp. 30–33, Nov. 2014, doi: 10.1109/ms.2014.139.
[82] P. Xiong and L. Peyton, "A model-driven penetration test framework for Web applications," in *Proc. 8th Int. Conf. Privacy, Secur. Trust*, Aug. 2010, pp. 173–180.
[83] P. Vats, M. Mandot, and A. Gosain, "A comprehensive literature review of penetration testing & its applications," in *Proc. 8th Int. Conf. Rel., Infocom Technol. Optim. (Trends Future Directions) (ICRITO)*, Jun. 2020, pp. 674–680.
[84] T. Borja, M. E. Benalcázar, L. V. Caraguay, and L. I. B. López, "Risk analysis and Android application penetration testing based on OWASP 2016," in *Proc. Int. Conf. Inf. Technol. Syst.* Cham, Switzerland: Springer, Feb. 2021, pp. 461–478.
[85] Z. Qu, S. Alam, Y. Chen, X. Zhou, W. Hong, and R. Riley, "DyDroid: Measuring dynamic code loading and its security implications in Android applications," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2017, pp. 415–426.
[86] S. Y. Yerima, M. K. Alzaylaee, and S. Sezer, "Machine learning-based dynamic analysis of Android apps with improved code coverage," *EURASIP J. Inf. Secur.*, vol. 2019, no. 1, Dec. 2019, pp. 1–24, doi: 10.1186/s13635-019-0087-1.
[87] S. Bojjagani and V. N. Sastry, "STAMBA: Security testing for Android mobile banking apps," in *Adv. Signal Process. Intell. Recognit. Syst.* Cham, Switzerland: Springer, 2016, pp. 671–683.
[88] A. Alanda, D. Satria, H. A. Mooduto, and B. Kurniawan, "Mobile application security penetration testing based on OWASP," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 846, no. 1, May 2020, Art. no. 012036.
[89] S. Aymaz, T. Cavdar, S. Aymaz, and E. Ozturk, "An analysis of load balancing strategies with wireshark in software defined networks," in *Proc. Int. Conf. Artif. Intell. Data Process. (IDAP)*, Sep. 2018, pp. 1–5.
[90] P. Shi, F. Qin, R. Cheng, and K. Zhu, "The penetration testing framework for large-scale network based on network fingerprint," in *Proc. Int. Conf. Commun., Inf. Syst. Comput. Eng. (CISCE)*, Jul. 2019, pp. 378–381.
[91] B. Mburano and W. Si, "Evaluation of Web vulnerability scanners based on OWASP benchmark," in *Proc. 26th Int. Conf. Syst. Eng. (ICSEng)*, Dec. 2018, pp. 1–6.
[92] M. B. Seyyar, F. Ö. Çatak, and E. Gül, "Detection of attack-targeted scans from the apache HTTP server access logs," *Appl. Comput. Informat.*, vol. 14, no. 1, pp. 28–36, Jan. 2018, doi: 10.1016/j.aci.2017.04.002.

[93] O. Aslan and R. Samet, "Investigation of possibilities to detect malware using existing tools," in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2017, pp. 1277–1284.
[94] T. Jain and N. Jain, "Framework for Web application vulnerability discovery and mitigation by customizing rules through ModSecurity," in *Proc. 6th Int. Conf. Signal Process. Integr. Netw. (SPIN)*, Mar. 2019, pp. 643–648.
[95] *OSSTMM 3—The Open Source Security Testing Methodology Manual*. Accessed: Nov. 21, 2019. [Online]. Available: https://www.isecom.org/OSSTMM.3.pdf
[96] (2019). Accessed: Sep. 3, 2019. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
[97] *OWASP Mobile Security Project—OWASP*. Owasp.Org. Accessed: Oct. 28, 2019. [Online]. Available: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#Top_Ten_Mobile_Risks
[98] *Technical Guide to Information Security Testing and Assessment*. Accessed: Sep. 4, 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf
[99] 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*, Apr. 2009, pp. 346–355.
[100] L. Clapp, O. Bastani, S. Anand, and A. Aiken, "Minimizing GUI event traces," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, Nov. 2016, pp. 422–434.
[101] Š. Packevičius, A. Ušaniov, Š. Stanskis, and E. Bareiša, "The testing method based on image analysis for automated detection of UI defects intended for mobile applications," in *Proc. Int. Conf. Inf. Softw. Technol.* Cham, Switzerland: Springer, Oct. 2015, pp. 560–576.
[102] F. Palma, N. Realista, C. Serrao, L. Nunes, J. Oliveira, and A. Almeida, "Automated security testing of Android applications for secure mobile development," in *Proc. IEEE Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Oct. 2020, pp. 222–231.
[103] R. Mahmood, N. Mirzaei, and S. Malek, "EvoDroid: Segmented evolutionary testing of Android apps," in *Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, Nov. 2014, pp. 599–609.
[104] R. R. Asaad, "Penetration testing: Wireless network attacks method on Kali Linux OS," *Acad. J. Nawroz Univ.*, vol. 10, no. 1, pp. 7–12, 2021.
[105] A. Avancini and M. Ceccato, "Security testing of the communication among Android applications," *Proc. 8th Int. Workshop Automat. Softw. Test (AST)*, May 2013, pp. 57–63.
[106] O.-E.-K. Aktouf, T. Zhang, J. Gao, and T. Uehara, "Testing location-based function services for mobile applications," in *Proc. IEEE Symp. Service-Oriented Syst. Eng.*, Mar. 2015, pp. 308–314.
[107] P. Zhang and S. Elbaum, "Amplifying tests to validate exception handling code: An extended study in the mobile application domain," *ACM Trans. Softw. Eng. Methodology*, vol. 23, no. 4, pp. 1–28, Sep. 2014.
[108] K. Yang, J. Zhuge, Y. Wang, L. Zhou, and H. Duan, "IntentFuzzer: Detecting capability leaks of Android applications," in *Proc. 9th ACM Symp. Inf., Comput. Commun. Secur.*, Jun. 2014, pp. 531–536.
[109] K. B. Dhanapal, K. S. Deepak, S. Sharma, S. P. Joglekar, A. Narang, A. Vashistha, P. Salunkhe, H. G. N. Rai, A. A. Somasundara, and S. Paul, "An innovative system for remote and automated testing of mobile phone applications," in *Proc. Annu. SRII Global Conf.*, Jul. 2012, pp. 44–54.
[110] (2019). *Vetting the Security of Mobile Applications*. Accessed: Aug. 9, 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-163r1.pdf
[111] P. Kong, L. Li, J. Gao, K. Liu, T. F. Bissyandé, and J. Klein, "Automated testing of Android apps: A systematic literature review," *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 45–66, Sep. 2018, doi: 10.1109/TR.2018.2865733.
[112] E. A. A. Vega, A. L. S. Orozco, and L. J. G. Villalba, "Benchmarking of pentesting tools" *Int. J. Comput., Elect., Automat., Control Inf. Eng.*, vol. 11, no. 5, pp. 602–605, 2017, doi: 10.5281/zenodo.1130587.
[113] F. R. Muñoz, E. A. A. Vega, and L. J. G. Villalba, "Analyzing the traffic of penetration testing tools with an IDS," *J. Supercomput.*, vol. 74, no. 12, pp. 6454–6469, Dec. 2018, doi: 10.1007/s11227-016-1920-7.
[114] Y. K. Lee, J. Y. Bang, G. Safi, A. Shahbazian, Y. Zhao, and N. Medvidovic, "A SEALANT for inter-app security holes in Android," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng. (ICSE)*, May 2017, pp. 312–323.
[115] D. C. Nguyen, D. Wermke, Y. Acar, M. Backes, C. Weir, and S. Fahl, "A stitch in time: Supporting Android developers in WritingSecure code," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1065–1077.
[116] A. K. Jha and W. J. Lee, "Analysis of permission-based security in Android through policy expert, developer, and end user perspectives," *J. UCS*, vol. 22, no. 4, pp. 459–474, 2016.
[117] A. Aldayel and K. Alnafjan, "Challenges and best practices for mobile application development," in *Proc. Int. Conf. Comput. Data Anal.*, May 2017, pp. 41–48.
[118] W. Ahmad, C. Kästner, J. Sunshine, and J. Aldrich, "Inter-app communication in Android: Developer challenges," in *Proc. 13th Int. Conf. Mining Softw. Repositories*, May 2016, pp. 177–188.
[119] C. M. Prathibhan, A. Malini, N. Venkatesh, and K. Sundarakantham, "An automated testing framework for testing Android mobile applications in the cloud," in *Proc. IEEE Int. Conf. Adv. Commun., Control Comput. Technol.*, May 2014, pp. 1216–1219.
[120] A. Pandey, R. Khan, and A. K. Srivastava, "Challenges in automation of test cases for mobile payment apps," in *Proc. 4th Int. Conf. Comput. Intell. Commun. Technol. (CICT)*, Feb. 2018, pp. 1–4.
[121] A. K. Jha, S. Lee, and W. J. Lee, "Developer mistakes in writing Android manifests: An empirical study of configuration errors," in *Proc. IEEE/ACM 14th Int. Conf. Mining Softw. Repositories (MSR)*, May 2017, pp. 25–36.
[122] M. Linares-Vasquez, C. Bernal-Cardenas, K. Moran, and D. Poshyvanyk, "How do developers test Android applications?" in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2017, pp. 613–622.
[123] (2019). *HTBridge/pivaa*. GitHub. Accessed: Sep. 16, 2019. [Online]. Available: https://github.com/HTBridge/pivaa
[124] (2019). *Yaazhini—Free Android APK & API Vulnerability Scanner | Vegabird*. Vegabird.com. Accessed: Oct. 14, 2019. [Online]. Available: https://www.vegabird.com/yaazhini/
[125] L. Xiao, Y. Li, X. Huang, and X. Du, "Cloud-based malware detection game for mobile devices with offloading," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2742–2750, Oct. 2017.
[126] S. Shah and B. M. Mehtre, "An overview of vulnerability assessment and penetration testing techniques," *J. Comput. Virol. Hacking Techn.*, vol. 11, no. 1, pp. 27–49, Feb. 2015.
[127] S. Bojjagani and V. N. Sastry, "VAPTAi: A threat model for vulnerability assessment and penetration testing of Android and iOS mobile banking apps," in *Proc. IEEE 3rd Int. Conf. Collaboration Internet Comput. (CIC)*, Oct. 2017, pp. 77–86.

**IKRAM UL HAQ** received the M.S. degree in software engineering from Bahria University, Islamabad, Pakistan, in 2013. He is currently a Computer Forensic Investigator with specialization in design of information systems. He has over 19 years of experience in the field of information systems, software development, team management, and project management. His main research interests include cyber security, big data, deep learning, and artificial intelligence. He is serving for Public Sector Organization.

**TAMIM AHMED KHAN** received the B.E. degree (Hons.) in software engineering from The University of Sheffield, U.K., in 1995, the M.B.A. degree in finance and accounting from Preston University, Islamabad, Pakistan, in 1997, the M.S. degree in computer engineering from CASE, Texila University, Pakistan, in 2006, and the Ph.D. degree in software engineering from the University of Leicester, U.K., in 2012. He is currently working as a Professor with the Department of Software Engineering, Bahria University, Islamabad. His research interests include service oriented architectures, e-learning, and software quality assurance.