IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 2, February 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.542

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 10, Issue 2, February 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1002006 |

Android Application Testing Using Fuzzing: Systematic Literature Review

Pranav Kulkarni, Vikrant Shinde

B.E. Student, Dept. of Computer, Trinity Academy of Engineering, Pune, India

B.E. Student, Dept. of Computer, Trinity Academy of Engineering, Pune, India

ABSTRACT: Android application markets are making a crucial shift in the manner software which was delivered to the end-users. Android applications face increasingly more security threats. The fuzzing technique can be used to uncover the security threats of applications. Fuzzing can be summarized similarly to the way toward sending irregular or invalid information as a contribution to a framework, to crash the framework and uncovering conceivable security vulnerabilities. Various research has been published in the android application fuzzing domain, while not many researches have addressed the security vulnerabilities in an android application by using external (another device used for operations) fuzzing techniques. However, current research doesn't address the android application fuzzing using an internal tool (or itself android device). The significance of the area, this paper seeks after two targets: to give a complete systematic literature review (SLR) of android application fuzzing, requirement's for android application fuzzing. This paper reviewed previous research study in android application fuzzing, also methods dependent on necessities in Kitchenham's systematic literature review guidelines. The systematic literature review result has shown the following deficiencies: Internal tool is not considered for fuzzing android application; and studies that are lacking in terms of requirement types. Furthermore, we proposed strong future directions of fuzzing for android application using internal tool. In particular, revealing threats of android applications by fuzzing techniques enables developers to build more secure applications and increases the mutual trust of its users.

KEYWORDS: Fuzzing, Android, Application, Fuzz, Testing, Internal tool, Android Application, Android Application fuzzing.

I. INTRODUCTION

Android application markets are making a crucial shift in the manner software which was delivered to the end-users. By giving a medium for reaching a large consumer market at a minimal cost, Android application markets have managed to level the software development field with compare to other desktop-based software. Small enterprises compete against major software development companies [1]. The result of this market share has exploded in recent years. According to recent statistics of Canalys [2], the market hit 369 million units in the fourth quarter of 2019.

Android is an open-source and free operating system based on Linux, it self-android is so powerful. Google play store having billions of app which are uses for different purposes. A core feature of Android is that one application component could use another application component element that belongs if the component is permitted for using it. Android Application is combination of different components Activity, Services, Broadcast receiver, Content provider [3]. Fig 1. Show Inter Component Communication (ICC). Inter Component Communication helps the android application to communicate with other applications and device drivers.

Android applications increasingly more security threats. Threats originate from numerous sources [4]–[6], input data which is unexpected one of the main reasons. Android applications uses, information originating from servers, records spared in nearby capacity, and tasks performed by users. These are three fundamental kinds of information [7]–[9]. Fuzzing technique which uses file as input is very common, and different type of fuzzing frameworks are available. American Fuzz Loop [10]

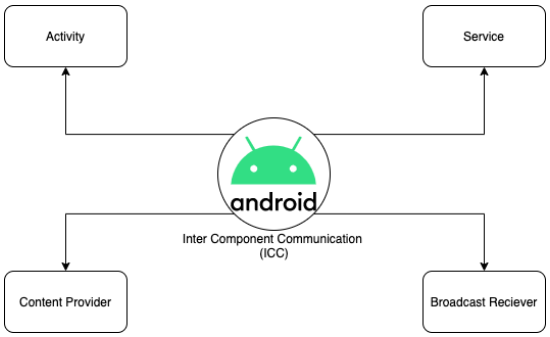
is a great tool for fuzzing system libraries, Peach[11] provides the graphical user interface for fuzzing different windows libraries. The main idea is fuzzing android applications for discovering potential security threats. In simple words, fuzzing is used to uncover applications code or logical problems by continuously sending invalid data from input point. Numerous studies on android application fuzzing only focuses on UI events on GUI events. Monkey [12] and MonekyRunner [13] are to fuzz tester provided by google. This tool only focuses on UI based fuzzing techniques.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 10, Issue 2, February 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1002006 |

Consistent with previous studies, android application fuzzing is very important in terms of security perspective. But very few studies are focused on android application fuzzing for finding security vulnerabilities. This systematic literature review study focusses on identifies current fuzzing techniques used for android application, what vulnerabilities detected by previous studies, methods, and requirements for android application fuzzing, Discussion on fuzzing android application, and future trends.



Different components urged to perform this systematic literature review. Another research studies has some limitations and drawbacks. Although different survey studies have addressed android application fuzzing by external methods. None have focused on fuzzing application through internal (Using an android application as a tool). Fuzzing of android applications can be used to found various types of vulnerabilities in Android applications. Many of the research uses an external method for fuzzing an android application. Thus, this inspired this systematic literature review research.

This systematic literature review empowers the recognizable proof of holes and difficulties that available in explicit research subjects, similar to Fuzzing of Android Application, therefore giving a complete review of the whole cutting edge in this area for researchers and practitioners.

To the best of our knowledge, there is no related literature review or survey summarizing the topic of android application fuzzing. We first attempted to meet this need through a comprehensive study. Strongly, we undertake a systematic literature review, and very carefully following the guidelines proposed by Kitchenham et. al. [14].

The rest of the paper is organized as follows. Section 2 describes our Research methodology. Section 3 Result and Discussion of research question on gathered research studies. Section 4 Future Directions for fuzzing for android application using internal tool and why it is important. Section 5 concludes this paper.

II. RESEARCH METHOD

This paper, Systematic literature review is used to manage the research studies procedure by Kitchenham et. al. [14]. Fig 2 shows the phases of methodology, methodology were developed on the procedure of Kitchenham et. al. [14]. This Systematic literature review having three phases: planning, conducting, and reporting. Each phase has several sub-phases. Planning phase involves the recognizing the need to lead this systematic literature review which defines research questions. The second phase, Systematic literature review directed. It has some sub-phases: search process, research study determination, and data extraction from studies. Reporting phase contains a discussion of outcome onresearch studies. Discussion of usage of Main phases and sub-phases are well explained in the sub-sections which are given below.

A. Planning

This systematic literature review underlines loop holes by analysing and identifying the importance of fuzzing android applications, requirements for fuzzing Android application, and existing techniques in wording of their fuzzing process,description, and exploitations to describe the vulnerabilities of android application. Some research questions of systematic literature review were based on the objective and requirements of this research.

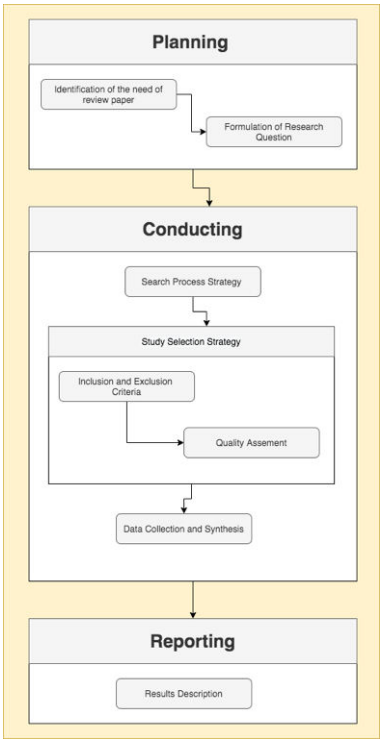
R.Q.1. What are the android application fuzz concerns?
R.Q.2. What are the Requirements for fuzzing Android application?
R.Q.3. What are the available models/methods/tool for fuzzing Android application?

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 10, Issue 2, February 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1002006 |

R.Q.4. What are the process, descriptions, limitations of each identified model/method/tool?



B. Conducting

Execution of conducting phase was done with help of following sub-phases: search process, research study determination, and data extraction from studies. The usage of this phase is showed below.

III. SEARCH PROCESS STRATEGY

An all-around characterized search process has a main job in gaining acceptable quality, and dependable outcomes. This systematic literature review, the search process significantly performed to remove the information and gather all relevant existing examination studies were the predetermined area of search keywords. The search keywords in this systematic literature review are related to the listed research questions. For This systematic literature review, we used serval electronic online libraries. These electronic online libraries are popular in research studies.

Electronic online libraries:

- ACM Digital Library
- IEEE Xplore
- Google Scholar
- ScienceDirect
- SpringerLink

2. Study Selection Strategy

After successful execution of conducting is depend upon well-defined search procedure, 240 research studies extracted from various online sources which are listed above. By sorting extracted research studies to distinguish the important research studies of the defined area of this research, Selection process were done with sub-phases: First is inclusion, exclusion criteria of an android application, and Second is quality assessment of selected studies. Inclusion-Exclusion phase are figured in this systematic literature review are depend on research questions. The inclusion and exclusion process show in TABLE 2. Gathered research studies sorted depend on the criteria of inclusion, exclusion. Only studies which focus on the android application fuzzing and android application or fuzzing and studies that incorporate in any event of the research questions by analysing their titles, abstracts, and keywords were included in this systematic literature review.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 10, Issue 2, February 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1002006 |

TABLE 2. Quality Assessment

Inclusion	Exclusion
Studies are written in English.	Studies are not written in English.
Studies propose or include methods for android application fuzzing	Studies do present sufficient technical details about the methods for android application fuzzing
Studies report issues, problems, or experience which point to android application fuzzing	Studies do not concern android application fuzzing
Studies are relevant to research questions based on title, abstract, and keywords of each studies.	Studies are not related to any defined research questions.

Non-English studies were excluded, and studies which only focusses on fuzzing applications, fuzzing, applications, androidi.e. irrelevant to our research area were excluded. The duplicate analysis was helps to remove duplicate studies and recent copies were included. Therefore, 74 research studies finalised after sorting results shows the implementation of inclusion and exclusion phase. To Conducting quality assessment phase, Questions for assessment are figured on research questions of this systematic literature review research area and guideline procedures of Kitchenam et al [14]. Title, abstract, full content of all collected research studies from selected studies are reviewed and figured checklist questions in TABLE 3. Question scored are defined: Yes=1, Moderately=0.5, and No=0; To ensure the findings of this research systematic literature review. Research study was not concerned if its total score is below 5, and the study selected if its total score more than 5. As a outcome, only 21 research studies finalised for primary studies of this systematic literature review. The primary studies and total scores of quality shows in TABLE 4. Number of studies collected and selected during each phase of this systematic literature review from each online electronic library are showed in TABLE 5.

TABLE 3. Quality Assessment

No	Question	Score
QA1	Are the aims started?	Yes=1, Moderately= 0.5, No=0
QA2	Is the study's well demonstrated?	Yes=1, Moderately= 0.5, No=0
QA3	Does the study focus on the specified domain of defined research questions?	Yes=1, Moderately= 0.5, No=0
QA4	Is the proposed solution compressively explained?	Yes=1, Moderately= 0.5, No=0
QA5	Do the results add critical findings to the literature?	Yes=1, Moderately= 0.5, No=0
QA6	Is the reporting clear and coherent?	Yes=1, Moderately= 0.5, No=0

TABLE 4. Scores for Result of Primary Studies

Ref.	QA1	QA2	QA3	QA4	QA5	QA6	Score
J. Burns [15]	1	1	1	1	1	1	6
H.Ye[16]	1	1	1	1	1	1	6
K.Yang[17]	1	1	1	1	1	1	6
A.Marchiry[18]	1	1	0.5	1	1	1	5.5
Yang[19]	1	1	0.5	1	1	1	5.5
W. Choi[20]	1	1	0.5	1	1	1	5.5
S.Hao[21]	1	1	0.5	1	1	1	5.5
R.Mahmood[22]	1	1	0.5	1	1	1	5.5
D.Amalditano[23]	1	1	0.5	1	1	1	5.5
S. Anand[24]	1	1	0.5	1	1	1	5.5
B.Liu[25]	1	1	0.5	1	1	1	5.5
Blanda [26]	1	1	1	1	1	1	6
Karim [27]	1	1	0.5	1	1	1	5.5

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 10, Issue 2, February 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1002006 |

3. Reporting

In reporting phase of systematic literature review, the Mendeley, EndNote, and other software's are utilized during the time of research information collection, references, and citations. Research information were separated and gathered dependent on his systematic literature review research paper questions, whereas each chosen primary study was essentially examine to obtain any information that can help with tending to the questions. Last data analysis phase, summed up proofs were gathered from the information assembled from the chose essential examination to respond to the recorded research questions.

TABLE 5. Studies shorted during each phase of systematic literature review from all electronic research libraries

Libraries	Search process strategy phase	Inclusion and exclusion Phase		Quality assessment phase	
	Collected	**Included**	**Excluded**	**Included**	**Excluded**
IEEE Xplore	23	10	14	6	18
ACM Digital Library	129	22	107	7	123
Google Scholar	46	33	13	7	34
ScienceDirect	8	3	5	0	8
SpringerLink	34	6	28	1	33

IV. DISCUSSION

A. R.Q.1 Why are the android application fuzz concerns?

Fuzzing has been used widely to discover vulnerabilities in software's and application [28]. Security researchers utilizing likewise fuzzing techniques to discovering bugs and vulnerabilities in programs, applications or software's. Purpose of crashing the system and revealing possible security vulnerabilities problems is need to be important in development of applications or softwares.

Android is one of the prevailing processing stages today since it has the main versatile market in advanced mobile phones and it is open source programming. Numerous individuals on the planet utilize these Android applications every day. A product with such a huge client base should be extremely vigorous and secure, in any case, even a few imperfections may prompt huge expenses. However, Android is vulnerable for many reasons. Two reasons which we focus on,

Reason 1. Android platform versions coexisting in the market from the newest version to the old ones [35].

Reason 2. Android applications are mostly written in Java or Katlin programming language with Android APIs (Application Programming Interfaces) are used for communicating with other servers. Android is Google's open-source platform for mobile devices, and it provides the APIs necessary to develop applications for the platform in Java [37].

Fuzzing can be considered; it is described as a black-box software or application testing technique to uncover the security vulnerabilities. Exceptionally at broad level, a meaning of fuzzing can be summarized just like the way toward sending arbitrary or invalid information as a contribution to a framework, to crash the framework and uncovering conceivable security vulnerabilities or unwavering quality issues. In the last decade, fuzzing has gradually developed, they have gained more popularity among security experts or researchers. Thus, we decided to do this systematic literature review on android application fuzzing. As of our best knowledge none of studies are done the systematic literature review on android application fuzzing.

B. R.Q.2 What are Requirements for fuzzing Android applications?

Fuzzing Android application is a unique task. TABLE 6 shows that the requirements of fuzzing android applications. Firstly, we classified into three broad categories Android Component, Test Fields, and Test Types. Android Application is consists of different components that are Activity, Service, Broadcast Receiver, and Content Provider [37]. Activity is the main interface on which users can interact with apps [16], [17], [35], [38], [18]–[24], [30]. Services provide essential support and fundamental functionalities for user android applications [25]. Binder component helps the android applications to perform calls into the codes of system services which were provided by system-level processes that is "system server" [39]. These are commonly found where applications want to register for system

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 10, Issue 2, February 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1002006 |

application events or actions. All registered receivers for an event are notified by the Android runtime once this event happens [40].

This requirement are general requirements for fuzzing any android application. However, the complexity of these prerequisites and issues increment when verities in test fields and test sorts of information. Application support different fields action, mime type, and URI (Unified Resource Identifies). Action in the android application is used to calling different system application intent. It is also use by other apps to invoke methods in applications. Action is the most crucial part in android applications. MIME-Type specifies the media type of the input data in application. Android application support many media type (e.g. text, pdf, mp4, etc.). URI (Unified Resource Identifies) is use to identifies different resource protocol. URI in the form of "scheme://host: port/path". Scheme are of two type one is predefined i.e http, https, file, ftp etc. and another is self-constructed i.e. fb, twitter etc. Testing types are defining what type of input data is sending to the app for fuzzing. In previous studies, Null, Random, Semi-Valid are types of data uses to fuzzing the android application.

TABLE 6. Requirements of Fuzzing android application

Requirement		Descriptions
Android Components	**Activity**	Activity is the main interface on which users can interact with[16], [17], [35], [38], [18]–[24], [30]
	Service	Services provide essential support and fundamental functionalities for user apps [25].
	Broadcast Receiver	Thesearecommonlyfoundwhereapplicationswanttoregisterforsystemorapplicationsevents or actions[40]
Testing Fields	**Action**	Testing with different action type fields encounter major vulnerabilities in applications[30], [35], [38]
	MIME Type	MIME specifies the media type of the data input.[16], [30]
	URI	URI is in the form of "scheme://host: port/path". [30]
Testing Types	**NULL**	Sending null data to find out the NullPointerException [17], [35], [38]
	Random	Random data added in inputs for fuzzing applications. [16], [21], [28], [30], [35]
	Semivalid	SemiValid data is generated from application behavior by extracting application code. [18]– [23], [30]

C. R.Q.3 What are the available models/methods/tool for fuzzing Androidapplications?

This formulated question mainly focuses only on what is available model/methods/tools which are available for fuzzing android applications.

Multiple researchers build the tools for fuzzing android application but most of themare external tools thatusehigh-performance computing. Methods/tools/models which are previously introduced by the researcher mostly focus on GUI based fuzz testing. Such as [18]–[24] are mainly focus on UI events, touch screen events, and GUI testing activities. UI testing is not considered in the form of security vulnerabilities. Some related but not relevant studies are introduced by some researchers which focuses on fuzzing android device drivers.[26]–[28], [33] researchers fuzz the android device drivers such as media frameworks. Actual studies that are focusses on android application fuzzing to find security vulnerabilities are [16], [17]

Multiple studies have been investigated to fuzzing techniques to find out different vulnerabilities in the android application. This paper reviewed the research studies and classified the methods interms of their android components, tested fields, test types, and vulnerabilities. Vulnerabilities are which are tested by each model. TABLE 7 Shows the detailed discussion. [16], [17], [30], [32], [35], [38] studies only focus on security vulnerabilities of android applications. Some GUI testing models also added in this study because of their process which is similar to security fuzzing, suchas[18]–[24].

D. R.Q.4 What are process, descriptions, limitations of each identified models/method/tool?

Many researchers present many models, but most of them are focusing on GUI based fuzzing techniques. Only a few models are uses fuzzing for finding security vulnerabilities is android applications. This paper only focuses on the fuzzing of android applications to find security vulnerabilities. Thus, we found [15], [17], [16][18], [30], [32], [35], [38] studies which focuses on security threats.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 10, Issue 2, February 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1002006 |

IntentFuzzer [15], the first Intent fuzzing tool for testing Intent vulnerability of Android application. It's an Android application, which use to fuzz other applications. IntentFuzzer collect information on installed applications and their Intent filters through Android API. It fuzz using Null data input and activity components only.
IntentFuzzer [17], combined a static analysis with random fuzzing to test Android apps dynamically. CFG analysis were employed to extract the structure of Intents that each target component expects. CFG uses the path-insensitive and inter- procedural. The analysis traverses Dalvik bytecode instructions to collect calls to Intent's getter or setter methods, and calls to their respective bundle objects, starting at entry point of each component's which is onCreate method in Activity. Method of this type of calls use a specific string which is denoted by key. It is use to extract extra data from Intents whereas data type is encoded in the same methods
DroidFuzzer [16], focused on the data field of Intents being set with malformed audio and video files only for Activity type components. Based on the extracted URI and MIME data type information from an Android app configuration file called AndroidManifest.xml. It built pieces of abnormal audio and video data for testing Activities. The tool is equipped with a dynamic crash monitoring module that is capable of detecting Activity crashes and native code crashes. DroidFuzzer uncovered bugs such as consumption of resources, ANR (Android Not Responding), and crashes from not dealing with malformed audio and video files well, rather than bugs resulting from the Intent field missing or incorrect types of Intent field values.
AFFH [30], the Study carried out vulnerabilities in the android application by fuzzing network traffic HTTP/HTTPS of android application. The practical study carried out vulnerabilities are android application not responding due crashes caused by JSON data exception, HTML content replacement, and URL redirection. This are major vulnerabilities found by this tools AFFH uses middle man technology to intercept the HTTP/HTTPS traffic. Model require some human interaction to exploit the security threats. So, the level of threat is low in terms of attacker perspective.

GFuzz [32], Focused on fuzzing the android application installation process with help of Dexfile. *classes.dex*file contains the android application source code. Study focuses on fuzzing the android system library by mutating data into the dex file. Apps are need to decompile to get of Dexfiles. Similar tools are also added in the studies which are focuses on installation fuzzing.
Hwacha [35], Android Application fuzzing tool for detecting Intent vulnerabilities of Android apps causing the robustness problem. Authors introduced Intent specification language to describe the structure of Intent. The study also implemented an LCS-based algorithm to sort duplicates entry in report function.
ICCFuzzer [38], is another interesting tool to uncover crashes by Null reference exception, Intent spoofing, Intent hijacking, and data leak by path-insensitive inter procedural CFG static analysis. It was applied to Android apps from DroidBench (https://github.com/secure-software-engineering/DroidBench) and Google Play. The number of vulnerabilities detected with this tool was compared with by IntentFuzzer [17]

TABLE 7. Fuzzing Model Analysis concerning vulnerabilities

Ref	**Components**			**Vulnerabilities**
	Android Component	Tested Fields	Test type	
[15]	Activity	None	Null	NullPointerExtection
[17]	Activity	None	Null	NullPointerExtection, DOS, Application not responding.
[16]	Activity	MIME-type	Random	App crash, ANR, Consumption of Resources
[28]	None	None	Random	Segmentation Fault, Bufferover flow
[26]	None	None	None	Segmentation Fault, Bufferover flow
[39]	None	None	None	Segmentation Fault, Bufferover flow
[30]	Activity	Action, MIME type, URI	Random, Semi-valid	Application unresponsiveness crashes caused by JSON data exception, HTML content replacement, and URL redirection.
[18]	Activity	None	Semi-valid	No vulnerabilities carried out, Only UI events fuzzing
[19]	Activity	None	Semi-valid	No vulnerabilities carried out, Only UI events fuzzing
[20]	Activity	None	Semi-valid	No vulnerabilities carried out, Only UI events fuzzing
[21]	Activity	None	Random	No vulnerabilities carried out, Only UI events fuzzing

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 10, Issue 2, February 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1002006 |

[22]	Activity	None	Semi-valid	No vulnerabilities carried out, Only UI events fuzzing
[23]	Activity	None	Semi-valid	No vulnerabilities carried out, Only UI events fuzzing
[24]	Activity	None	Semi-valid	No vulnerabilities carried out, Only UI events fuzzing
[25]	Service	None	None	Summation fault.
[27]	None	None	None	No vulnerabilities carried out, Only UI events fuzzing
[29]	None	None	None	No vulnerabilities carried out, Only UI events fuzzing
[31]	None	None	None	No vulnerabilities carried out, Only UI events fuzzing
[32]	None	None	None	No vulnerabilities carried out, Only UI events fuzzing
[34]	None	None	None	No vulnerabilities carried out, Only UI events fuzzing
[35]	Activity	Action	Null, Random	NullPointerExtection, DOS, Application not responding.
[38]	Activity	Action	Null	NullPointerExtection, DOS, Application not responding.

V. FUTURE RESEARCH DIRECTIONS

Based on results of this systematic literature review and observations, we present the following research directions of fuzzing android application.

Fuzzing using internal tool/method: Security researchers utilizing fuzzing techniques to discovering bugs and vulnerabilities in android applications. Purpose of crashing the system and revealing possible security vulnerabilities problems is need to be important in development of applications or software's. Fuzzing has been used widely to discover vulnerabilities in software's and application [28].

Application developers are mean to be only develop applications. Developers are not that much aware about security testing of applications. It needs to be developed simple and understandable fuzzing testing for developers. Because of this, we are proposing the fuzzing for android application using internal tool/method. Internal tool can be referred as an android application. Android itself a powerful operating system, one application can be used to test another application like desktop application.

However, only one approach focuses on fuzzing of android application using internal tool. IntentFuzzer [15], is only approach which uses the internal method techniques. But this tool has limitations that it can be fuzz only the activities of android application by using only NULL data as input.

In this systematic literature review, we are carried out many different approaches of fuzzing android application. We classified the android components, testing fields, and test type for fuzzing android application with respect to their approach and vulnerabilities found during their tests.

VI. CONCLUSION

This study investigates the current state-of-art on android applications fuzzing. During systematic literature review, 21 Studies are selected, that were published in major conferences, workshops, and journals in the area software engineering, programming language, and security domain. This Paper answers research questions (RQs) that encounter during the research study. Studies that are focused on android application fuzzing and find security threats are uses external devices for their methods. Such devices are very costly and required higher performance capabilities. None of the studies have been addressed the fuzzing using an internal tool or using an android application. This systematic literature review intensively reviewed the various research papers in the related area of android application fuzzing. Previous research studies have not addressed the fuzzing using an android device. We proposed the arrangement of prerequisites for android application fuzzing that can be utilized as the premise of such a model. In particular, revealing threats of android applications by fuzzing techniques enables developers to make more secure android applications and

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 10, Issue 2, February 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1002006 |

increases the mutual trust of its users. Using Android as a tool for fuzzing will helps the developer for finding security threats to the android application.

REFERENCES

1. S.Malek,N.Esfahani,T.Kacem,R.Mahmood,N.Mirzaei,andA.Stavrou,"Aframeworkforautomatedsecurity testing of android applications on the cloud," *Proc. 2012 IEEE 6th Int. Conf. Softw. Secur. Reliab. Companion, SERE-C 2012*, pp. 35–36, 2012, doi:10.1109/SERE-C.2012.39.
2. Canalys, "Canalys Newsroom- Global smartphone market Q4 and full year 2019," *Canalys*, 2020. https://www.canalys.com/newsroom/canalys-global-smartphone-market-q4-2019 (accessed May 16,2020).
3. J. Liu and J. Yu, "Research on development of android applications," *Proc. - 2011 4th Int. Conf. Intell. Networks Intell. Syst. ICINIS 2011*, pp. 69–72, 2011, doi:10.1109/ICINIS.2011.40.
4. J.Ma,S.Liu,Y.Jiang,X.Tao,C.Xu,andJ.Lu,"LESdroid:AtoolfordetectingexportedserviceleaksofAndroid applications," *Proc. - Int. Conf. Softw. Eng.*, pp. 244–254, 2018, doi:10.1145/3196321.3196336.
5. Q. Zeng, L. Luo, Z. Qian, X. Du, and Z. Li, "Resilient decentralized android application repackaging detection using logic bombs," *CGO 2018 - Proc. 2018 Int. Symp. Code Gener. Optim.*, vol. 2018-Febru, pp. 50–61, 2018, doi:10.1145/3168820.
6. A.HamidrezaandN.Mohammed,"Permission-basedAnalysisofAndroidApplicationsUsingCategorizationand DeepLearningScheme,"*MATECWebConf.*,vol.255,p.05005,2019,doi:10.1051/matecconf/201925505005.
7. L. Zhang *et al.*, "Invetter: Locating insecure input validations in android services," *Proc. ACM Conf. Comput. Commun. Secur.*, pp. 1165–1178, 2018, doi:10.1145/3243734.3243843.
8. L.CasatiandA.Visconti,"TheDangersofRooting:DataLeakageDetectioninAndroidApplications,"*Mob.Inf. Syst.*, vol. 2018, 2018, doi:10.1155/2018/6020461.
9. S. Kelkar, T. Kraus, D. Morgan, J. Zhang, and R. Dai, "Analyzing HTTP-Based Information Exfiltration of Malicious Android Applications," *Proc. - 17th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 12th IEEE Int.Conf.BigDataSci.Eng.Trust.2018*,pp.1642–1645,2018,doi:10.1109/TrustCom/BigDataSE.2018.00242.
10. "american fuzzy lop." https://lcamtuf.coredump.cx/afl/ (accessed May 27,2020).
11. "Peach Fuzzer - Peach Tech." https://www.peach.tech/products/peach-fuzzer/ (accessed May 27,2020).
12. "UI/Application Exerciser Monkey | Android Developers." https://developer.android.com/studio/test/monkey (accessed May 27, 2020).
13. "monkeyrunner | Android Developers." https://developer.android.com/studio/test/monkeyrunner (accessed May 27, 2020).
14. Kitchenham and S. Charters, "Source: "Guidelines for performing Systematic Literature Reviews in SE", Kitchenham et al Guidelines for performing Systematic Literature Reviews in Software Engineering," 2007.
15. J. Burns, "Digging into droids. Exploratory AndroidTM Surgery," 2009. Accessed: Jun. 08, 2020. [Online]. Available: https://www.isecpartners.com.
16. H. Ye, S. Cheng, L. Zhang, and F. Jiang, "DroidFuzzer: Fuzzing the Android apps with intent-filter tag," ACM Int. Conf. Proceeding Ser., pp. 68–74, 2013, doi: 10.1145/2536853.2536881.
17. K. Yang, J. Zhuge, Y. Wang, L. Zhou, and H. Duan, "IntentFuzzer: Detecting capability leaks of android applications," ASIA CCS 2014 - Proc. 9th ACM Symp. Information, Comput. Commun. Secur., pp. 531–536, 2014, doi: 10.1145/2590296.2590316.
18. Machiry, R. Tahiliani, and M. Naik, "Dynodroid: An input generation system for android apps," 2013 9th Jt. Meet. Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng. ESEC/FSE 2013 - Proc., pp. 224–234, 2013, doi: 10.1145/2491411.2491450.
19. W. Yang, M. R. Prasad, and T. Xie, "A grey-box approach for automated GUI-model generation of mobile applications," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 7793 LNCS, pp. 250–265, 2013, doi: 10.1007/978-3-642-37057-1_19.
20. W. Choi, G. Necula, and K. Sen, "Guided GUI testing of android apps with minimal restart and approximate learning," Proc. Conf. Object-Oriented Program. Syst. Lang. Appl. OOPSLA, pp. 623–639, 2013, doi: 10.1145/2509136.2509552.
21. S. Hao, B. Liu, S. Nath, W. G. J. Halfond, and R. Govindan, "[KY] PUMA : Programmable UI-Automation for Large-Scale Dynamic Analysis of Mobile Apps Categories and Subject Descriptors," MobiSys '14, pp. 204–217, 2014.
22. R. Mahmood, N. Mirzaei, and S. Malek, "EvoDroid: Segmented evolutionary testing of Android apps," Proc. ACM SIGSOFT Symp. Found. Softw. Eng., vol. 16-21-Nove, pp. 599–609, 2014, doi: 10.1145/2635868.2635896.
23. Amalfitano, A. R. Fasolino, P. Tramontana, S. De Carmine, and A. M. Memon, "Using GUI ripping for automated

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 10, Issue 2, February 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1002006 |

testing of android applications," 2012 27th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2012 - Proc., pp. 258–261, 2012, doi: 10.1145/2351676.2351717.

24. S. Anand, M. Naik, M. J. Harrold, and H. Yang, "Automated concolic testing of smartphone apps," Proc. ACM SIGSOFT 20th Int. Symp. Found. Softw. Eng. FSE 2012, pp. 1–11, 2012, doi: 10.1145/2393596.2393666.
25. B. Liu, C. Zhang, G. Gong, Y. Zeng, H. Ruan, and J. Zhuge, "FANS : Fuzzing Android Native System Services via Automated Interface Analysis," USENIX Secur., no. 1, 2020.
26. Blanda, "Fuzzing Android: a recipe for uncovering vulnerabilities inside system components in Android."
27. Karim, F. Cicala, S. R. Hussain, O. Chowdhury, and E. Bertino, "Opening Pandora's box through Atfuzzer: Dynamic analysis of at interface for android smartphones," ACM Int. Conf. Proceeding Ser., pp. 529–543, 2019, doi: 10.1145/3359789.3359833

INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 7.542

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

Scan to save the contact details