International Journal of Engineering Research And Management (IJERM)
ISSN : 2349- 2058, Volume-03, Issue-07, July 2016

Security Framework for Model Based and Fuzzing Testing in Mobile Application

Vishruti Desai, Dr. Vivaksha Jariwala

***Abstract*— In today's ubiquitous world, mobile phones get an important place in people life, without which they feel handicapped or uncomfortable. When an application is installed on the user's device that is prone to a vulnerability of the device as well as application files. Information flow using mobile devices are also not in encrypted for most of the time. Because of this reason, Security is more important to user as well as to the application developer. Mobile application security, the process to verify vulnerabilities issues, is the main concern in the development of mobile application. In this paper, we propose the framework to test and verify mobile application, to be acted as an attacker, to prevent the application and user details to be misused. We suggested combining the approach of model based and fuzzing testing approach to find the security defects and threats from the developer and third party verification of user identification.**

***Index Terms*—Fuzzing testing, model based testing, mobile application, security testing.**

I. Introduction

Smart device usage is growing day by day. In recent years, more advancement in technology and mobile computing evolved. People install and download apps which are helpful to them to get connected with world, social media as well as professionally. However, attackers can easily be eavesdropping information and inject various malware activities in form of attack. Such issues must be detected or prevented using different techniques.

Software testing is as old as the hills in the history of the digital world. It is as much important to uphold the quality of measuring software [1]. In [2], Software testing reveals the only presence of defects, but not able to find an absence of defects. Software testing consists of the dynamic verification of a program which provides expected behaviours with actual behaviours on a finite set of test cases, derived from the infinite execution input domain, so called dynamic testing. The application which need be tested is called Application Under Test (AUT). Based on the objectives, accessibility and scope of testing, it can be classified and mapped on three dimensions' [3] as shown in Fig. 1.

In this paper, we first discuss the way third party verification is to be taken by the application developer. Model based

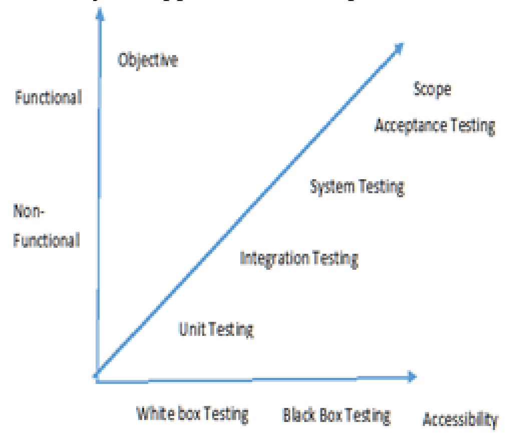

Figure 1: **Three Dimensions of testing [3]**

testing and Fuzzing testing functionality is also briefly described. Then different threats suggested by OWASP (Open Web Application Security Project) for mobile application. In last section, our proposed approach as a tester for detecting vulnerability in testing phase of mobile application. In the last section, we discuss issues to be attended for prevention of attacks.

II. Background

Any mobile applications are on security risk edge because of two reasons, one is security flaw originate at the time of development and the other way application is cloned with extra features that run in the background and work as malicious code.

A. *User third Party Authorization*

As suggested in [4], OAuth 2.0 authorization framework (RFC 6749), describes the complete procedure for authorization of user using a third party. The aauthorization code grant flow is cryptographically secure as it uses transport layer security (TLS) [14].

As shown in fig. 2, gives steps as it is mentioned.

Step A: As an initialization step, client request sends to resource owner for authentication, that can be verified directly from resource owner or indirectly verify by the authorization server as an intermediate.

Step B: After valid authorization, client receives an access grant credential that depends on type of method client uses or authentication as well as grant types supported by the server.

Step C: In the next step, client requests for an access token together with grant permission from the third party authorization server.

Step D: As an acknowledgement for step 3, sever gives access

Manuscript received July 22, 2016.
Vishruti A. Desai, Computer Engineering Department, C.K. Pithawalla College of Engineering and Technology, Surat, India
Dr. Vivaksha Jariwala, Associate Professor, Information Technology Department, Sarvajanik College of Engineering and Technology, Surat,

```
+--------+                               +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(C)-- Authorization Grant -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |                               |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

Figure 2: User Authorization flow in OAuth 2.0[4]

token of resource owner, if valid grant type and credential.
Step E: Afterwards any access by user from the resource server happen through access token only.
Step F: With valid access token, permission is given to client for resource utilization.

B. *Model Based Security Testing*

Model-based security testing (MBST) is an emerge range in research area of software testing and especially relies on system models to test. Mobile application designs meet its non-functional requirements in the form of security requirements. As mentioned in [5], MBST is basically devoted to the systematic and efficient security based documented specification. That documentation contains test objectives, test cases and test oracles, as well as to their automated or semi-automated generations. MBST is used for generating test cases (events) by considering different model representation using an algorithm or graph theory or state machine diagram, so it can easily be applied to generate it. Different models generate different path coverage. We need to consider the technique, that gives all possible paths in the model.

C. *Fuzzing Testing*

Fuzzing needs to be integrated with another testing approach to be easily applicable within application context and organization boundary. The author stated in [6], Fuzzing is a form of vulnerability testing that can be used to test any kind of application (web, mobile). It is excellent for exposing very serious vulnerabilities. The idea behind it is repeatedly send malformed data to the application in the request and response (or not) that it causes it to crash. Fuzzing does not require to involve source code, it is easy for attacker to inject vulnerability in the application. There are various forms of fuzzing: mutation-based, generation-based and behavior-based. Each one has merit and demerits, based on application specific can be chosen. In focus, Behaviour fuzzing finds flaws in the system design model and vulnerabilities within it that are not only exposed by applying invalid input data.

III. RELATED WORK

In [7], as per OWASP provided mobile application threats are Insecure Data Storage, Weak Server-Side Controls, Insufficient Transport Layer Protection, Client Side Injection, Poor Authorization and Authentication, Improper Session Handling, Security Decisions via Untrusted Inputs, Side Channel Data Leakage, Broken Cryptography, Sensitive Information Disclosure. We need to take any of the security problem for user information protection.
As defined in [8], various taxonomy for model based testing with different filter and evidence criteria are provided. As a filter criterion, system security model, environmental model, and explicit test selection are listed. For evidence, maturity of evaluated system maturity, measures, and level are also listed.
As author in [9] stated that OWASP of mobile security consider end user device security as well as server side infrastructure security. The intruder has access to the root with privileges that is more vulnerable as security risk [9].
In [10] authors specify, there aren't any global standard for mobile security guidelines which must meet for mobile devices. They only consider security in physical, transport and application layers of OSI. Apart from no standard, some security threats are identified by the National Institute of Standards and Technology (NIST) that includes device loss, virus and malware through USB devices, wireless ports and many other ways, and lastly while accessing spam.
Now in [11] author proposed, security in software is not only correctness and completeness of security function. As it is verified in the requirement phase but its more than functionality and performance of the system. Even different software security testing techniques, Code reviews, Automated static analysis, Binary code analysis, Fuzz testing, Source and binary code fault injection, Risk analysis, Vulnerability scanning, Penetration testing are listed.
In the survey report [12], authors have conducted the interview with industry people by questioning. It is used in website development, communication, embedded software and many more. Automatic test case generation, reusability of model components listed as benefits of it. Still, many industry experts are not skill for this type of testing because of limited knowledge in model based. Tools are not user friendly. Apart from that benefits are reduce the cost, efficient, more accurate.
Two RFC [4] and [13], OAuth Protocol, basically focused on architectural protocol specification and design in related with security.
HTTP Fuzzer [15], WebScarab [16], and NoTamper [17], are testing tools which accept pre-request fuzzing, fail to observe different responses for the requests [18] [19]. Black box testing just takes protocol based information and does not give full coverage. Apart from this Fuzzing works, model-based testing (MBT) give full coverage including all paths.
It is stated in [20] , to find defects with manual testing, MBT give better performance as it is mostly used in software testing [21], for web application, client server and embedded systems. While previously implemented various MBT [22] [23] [24], consider correct model is available during testing.

IV. OUR PROPOSED APPROACH FOR FRAMEWORK

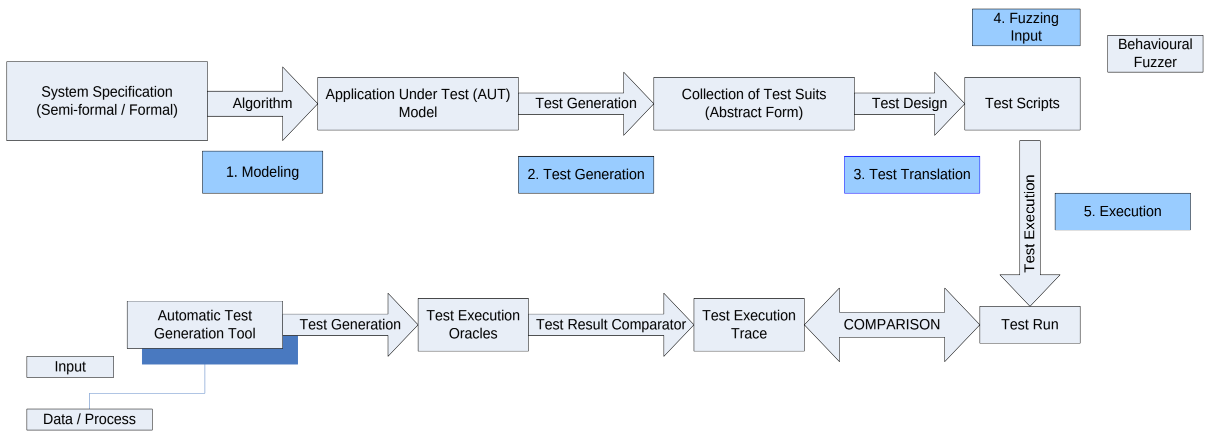

Figure 3: Steps for proposed framework

The way software application is delivered to end users, downloaded in markets create the false sense of security. Traditional security approach for software applications in commercial purpose are very different than mobile application. Industrial software development practice, security is not the only issue for mobile application, but it leads to compromises of millions of user's privacy and devices information.

Novel research challenge arises in software testing area. One is to appease model-based derivation of test cases with evolving dynamically modern systems. Another is to select and use runtime environment data collected from real usage when application deploy in mobile device. During requirement stage, design models and graphical representation are used in Software Requirement Specification (SRS) to represent the system. In SRS, system specification is written in semiformal or formal languages. In our proposed framework, we believe correctness and completeness of the system specification with user requirements.

As fig. 3 describes the basic blocks and the steps needed to use combining the approach of model based and fuzzing. Here we consider the advantages of both. As model based testing, we represent our system (an application) in the form of model that gives all path coverage for generating test cases. In fuzzing, generate invalid input data as well as change event sequence necessary to generate the valid response from the server as well as third party authentication [11]. Our proposed framework works on following steps and finally generates the report about negative response and security threats in the system. We assume tester will work as an attacker for the application to find the loophole of the application. With the procedure of our framework that works with the mobile application and check the security properties of it.

In fig. 3, it gives a flow of our proposed testing framework with actual testing steps taken while tester checks any application for verification.

1.Modelling: Give input as a system specification, must be taken in the form of semi-formal or formal specification, is given to the algorithm. The algorithm finds all the possible path coverage to generate different sequence of execution.

2.Test generator: Various paths generated in the previous step, together make one test suits in abstract form.

3.Test translation: As a translation, derived test suit is generated and well-formed written as a various test scripts. It covers all test condition and request response sequence.

4.Fuzzing input: Now input to test script will be generated as fuzzing input. Inputs are invalid form of data or random sequences; those are not same as it is expected for the application to test the validation.

5. Execution: Test scripts with fuzzed input are executed on AUT and the report will be generated.

Finally, generated report will be compared with expected test response, or validation of input with performance measure of the system. If any flaw or security threats will find, then system design gets a redesign and adjusted as per needs.

V. CONCLUSION AND FUTURE WORK

With functional requirement verification, software quality measures are also important. As a non-functional requirement, quality parameter of software include reliability, throughput and security are closely coupled with each other. In the current world, security is the main concern while developing the application. Any flaws in the application can be exploited by intruders or attacker as a security hole within the system. Growing Internet, mobile security issues become more challenging for preserving user privacy of data as well as information residing on mobile devices.

In this, we consider two different testing approaches during testing of mobile application. Both techniques are used for web based software. We try to frame them together by considering the limitations. The critical mobile applications need security framework with security properties (confidentiality, authentication, no repudiation, integrity) need to be considered against malicious attack.

The main aims of our framework, to provide identification of system flaw that leads to violation of security properties as well as validating effectiveness of valid input to the system. After finding the security flaw, correct the code and redesign or adjust the system model as per validation.

References

[1] Luo, Lu. "Software Testing Techniques: Technology Maturation and Research Strategy." Class Report for (2001).

[2] Bourque, Pierre, and Richard E. Fairley. Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press, 2014.

[3] P. Amman, J. Offutt, Introduction to Software Testing, Cambridge University Press, Cambridge, UK, 2008.

[4] D. Hardt, ." RFC6749: The OAuth 2.0 authorization framework". 2012

[5] Schieferdecker, Ina, Juergen Grossmann, and Martin Schneider. "Model-based security testing." arXiv preprint arXiv:1202.6118 (2012).

[6] Aaron Wishnick, Ming Chow, "Fuzzing an iOS Application", Tufts University, Computer Science, Introduction to Computer Security, December 13, 2013

[7] OWASP Foundation, OWASP Testing Guide v4, https://www.owasp.org/index.php/OWASP_Testing_Project accessed March 11, 2015.

[8] Michael Felderer, Philipp Zech, Ruth Breu, Matthias Buchler, Alexander Pretschner, "Model-Based Security Testing: A Taxonomy and Systematic Classification", SOFTWARE TESTING, VERIFICATION AND RELIABILITY,2014, pg no. 1-29

[9] Nicholas Penning, Michael Hoffman, Jason Nikolai, Yong Wang. "Mobile Malware Security Challeges and Cloud-Based Detection", 2014.

[10] Y. Cifuentes, L. Beltrán, L. Ramírez , "Analysis of Security Vulnerabilities for Mobile Health Applications" ,World Academy of Science, Engineering and Technology International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering Vol:9, No:9, 2015

[11] Abdullah Saad AL-Malaise AL-Ghamdi," A Survey on Software Security Testing Techniques", International Journal of Computer Science and Telecommunications, Volume 4, Issue 4, April 2013, pg no 14-18.

[12] Robert V. Binder, Anne Kramer, Bruno Legeard, 2014 Model-based Testing User Survey: Results, 2nd User Conference on Advanced Automated Testing (UCAAT) in Münich,October 2014.

[13] T. Lodderstedt, M. McGloin, and P. Hunt. RFC6819: OAuth 2.0 threat model and security considerations. 2013.

[14] S. Chari, C. S. Jutla, and A. Roy., "Universally composable security analysis of OAuth v2.0." IACR Cryptology ePrint Archive, 2011

[15] R. Abela. HTTP Fuzzer. acunitex.

[16] OWASP. Fuzzing with WebScarab

[17] J. Jacky. "Pymodel: Model-based testing in Python". In Proceedings of the Python for Scientific Computing Conference, 2011.

[18] A. Doupé, L. Cavedon, C. Kruegel, and G. Vigna. "Enemy of the state: A state-aware black-box web vulnerability scanner" In USENIX Security, 2012.

[19] G. Pellegrino and D. Balzarotti, "Toward black-box detection of logic flaws in web applications" in NDSS, 2014.

[20] C. Schulze, D. Ganesan, M. Lindvall, R. Cleaveland, and D. Goldman. "Assessing model-based testing: an empirical study conducted in industry" in Companion Proceedings of the International Conference on Software Engineering.ACM, 2014.

[21] A. C. Dias Neto, R. Subramanyan, M. Vieira, and G. H. Travassos. "A survey on model-based testing approaches: a systematic review" In Proceedings of ACM international workshop on Empirical assessment of software engineering languages and technologies, 2007.

[22] J. Ernits, R. Roo, J. Jacky, and M. Veanes. "Model-based testing of web applications" using NModel. Springer, 2009.

[23] J. Ernits, M. Veanes, and J. Helander. "Model-based testing of robots with NModel" Proc. Microsoft Research, 2008.

[24] G. Maatoug, F. Dadeau, and M. Rusinowitch.," Model-based vulnerability testing of payment protocol implementations" in HotSpot'14-2nd Workshop on Hot Issues in Security Principles and Trust, 2014.

Ms. Vishruti Desai, completed M.Tech. from VJTI, Mumbai, currently pursuing my Ph.D. Her research area includes software engineering, testing, mobile security, mobile computing.

Dr. Vivaksha Jariwala completed her Ph.D. from Sardar Vallabhbhai National Institute of Technology, Surat. Her research area includes information security, wireless sensor networks and software engineering.