ISSN: 2321-9653

INTERNATIONAL JOURNAL FOR RESEARCH IN APPLIED SCIENCE AND ENGINEERING TECHNOLOGY (IJRASET)

# Android Environment Security Through Penetration Testing Methodology

Sangita Dahiya[#1], Vinod Saroha

M.Tech(NS),BPSMVV Sonipat, *Asstt, Prof. in BPSMVV Sonipat*

***Abstract:*** ***The fluidity of application markets complicate Smartphone security. Although recent efforts have shed light on particular security issues, there remains little insight into broader security characteristics of Smartphone applications. This paper seeks to better understand Smartphone application security by studying tested its security through penetration testing. We introduce the ded decompiler, which recovers Android application source code directly from its installation image. Our analysis uncovered pervasive use/misuse of personal/ phone identifiers, and deep penetration of advertising and analytics networks. In this paper information about what security risks and attacks that are possible to execute towards a mobile device running Android will be presented. Possible attack scenarios are attacking the device itself, the communication between the device and a server and finally the server. Penetration testing is one of the oldest methods for assessing the security of a computer system. The idea behind penetration testing methodologies is that the penetration tester should follow a pre-scripted format during test as dictated by the methodology. A penetration testing methodology based on the research "4-step penetration planning" was proposed in this research***

***Keywords: Android, mobile device, penetration testing, Security, Vulnerability***

## I. INTRODUCTION

Smartphone growth and adaptation is increasing rapidly due to their rich and versatile functionality. The versatility and convenience of these devices took them an ahead from other apparently similar devices like PDAs (Personal Digital Assistants) or MIDs (Mobile Internet Devices). Nowadays, a Smartphone is not just used to talk; rather it gives functionality of a Pager, PDA (Personal Digital Assistants), MID, GPS, MP3 Player, etc., and provides a range of services like entertainment, electronic banking, reading e-books or attending office meetings online.Such a variety of services can only be delivered with the combination of strong compact hardware and fast reliable software including a good Operating System. Currently, the Android is one of the most popular open source operating systems for Smartphone's. It was originally developed by Google in 2005. Further development, the Android Open Source Project (AOSP) was established by Google and other members of Open Handset Alliance. Android is based on the Monolithic kernel (Modified Linux kernel) and contains all advanced features like multi-touch, video calling connectivity, multimedia messaging and web browsing. Several features and functions help to increase usage of data and services but also open the risk for introducing new vulnerabilities. According to a survey that was released in February 2011 by a customer intelligence firm,Market Force, 33% of the individuals don't have a Smartphone, 34 % intend to purchase one having an Android operating system in the upcoming six months. From these potential customers, 21% said that they would buy an IPhone,

ISSN: 2321-9653



12% said that they would buy a Blackberry and 25% did not decided, what to buy . This survey shows the increasing interest of potential customers in Android based Smartphones. Android Smartphones rapid growth and adaptation makes it more attractive for hackers. To protect against attacks, as many system vulnerabilities as possible should be found and patched on a forehand. To detect the vulnerabilities of a system, penetration testing is a very important tool which helps finding security holes in the system. A penetration test, occasionally called pentest, is a method of evaluating the security of a computer system or network by simulating an attack from a malicious source, known as a Black Hat Hacker or Cracker . The methodology for how to perform penetration tests is given by National Institute of Standards and Technology . Along with penetration testing, a general overview about the Android security mechanism is described to give the reader an idea of how it works.

## II. BACKGROUND

Mobile operating systems have been in use since the creation of the first mobile phone but those operating systems were targeted for specific devices. Most new devices used to come with improved operating systems but those improvements were often unnoticeable from the view of an ordinary user. With the advent of the Smartphone, the need for secure and robust multi-tasking mobile operating systems increased instantly.The development of hand-held operating systems began for devices like Smartphones, personal digital assistants (PDAs) and mobile internet devices (MIDs). Although it is not well-defined in the industry what a Smartphone really is, IBM Simon is considered to be the first Smartphone which had advanced features for its time where functionality such as Fax, Pager and PDAs as introduced. Modern Smartphones have much more advanced hardware and stronger operating system support. Nokia Corporation introduced the "Nokia Communicator Series", which is the most prominent milestone towards existing Smartphones. GEOS were initially used by Nokia in its communicator series and later in 2001 the Symbian OS was deployed in the latest model of the time which was the Nokia 9210 Communicator. Since then, Nokia with its Symbian OS led the Smartphone industry for the next ten years. Android, which was released in 2008, emerged as the major rival of Symbian OS and BlackBerry's RIM OS. According to Gartner statistics, in one year (2009 to 2010),Android based Smartphones increased their market share from 3% to 23% in the fourth quarter of 2010 . As the Smartphones become more and more popular, new application development and quick releases of new functionality is becoming the key factor for the growth of Smartphones.

Dalvik Virtual Machine: Android applications are written in Java, but run in the DVM. The DVMand Java bytecode run-time environments differ substantially

:*Application Structure*. Java applications are composed of one or more .class files, one file per class. The JVM loads the bytecode for a Java class from the associated .class file as it is referenced at run time. Conversely, a Dalvik application consists of a single .dex file containing all application classes. Figure 2 provides a conceptual view of the compilation process for DVM applications. After the Java compiler creates JVM bytecode, the Dalvik dx compiler consumes the .class files, recompiles them to Dalvik bytecode, and writes the resulting application into a single .dex file. This process consists of the translation, reconstruction, and interpretation of three basic elements of the application: the constant pools, the class definitions, and the data segment. A constant pool describes, not surprisingly,the constants used by a class. This includes, among other items, references to other classes, method names, and numerical constants. The class definitions consist in the basic information such as access flags and class names. The data element contains the method code executed by the target VM, as well as other information related to methods (e.g., number of DVM registers used, local variable table, and operand stack sizes) and to class and instance variables.

*Register architecture*. The DVM is register-based,whereas existing JVMs are stack-based. Java bytecode

can assign local variables to a local variable table before pushing them onto an operand stack for manipulation by opcodes, but it can also just work on the stack without explicitly storing variables in the table. Dalvik bytecode assigns local variables to any of the 216 available registers. The Dalvik opcodes directly manipulate registers, rather than accessing elements on a program stack.

*Instruction set*. The Dalvik bytecode instruction set is substantially different than that of Java. Dalvik has opcodes while Java has 200; however, the nature of the opcodes is very different. For example, Java has tens of opcodes dedicated to moving elements between the stack and local variable table.

*Constant pool structure*. Java applications replicate elements in constant pools within the multiple .class files, e.g., referrer

ISSN: 2321-9653



and referent method names. The dx compiler eliminates much of this replication. Dalvik uses a single pool that all classes simultaneously reference. Additionally,

dx eliminates some constants by inlining their values directly into the bytecode.

*Control flow Structure*. Control flow elements such as loops, switch statements and exception handlers are structured differently in Dalvik and Java bytecode. Java bytecode structure loosely mirrors the source code, whereas Dalvik bytecode does not.

*Ambiguous primitive types*. Java bytecode variable assignments distinguish between integer (int) and

single-precision floating-point (float) constants and between long integer (long) and double-precision floatingpoint (double) constants. However, Dalvik assignments (int/float and long/double) use the same opcodes for integers and floats, e.g., the opcodes are untyped beyond specifying precision.

*Null references*. The Dalvik bytecode does not specify a null type, instead opting to use a zero value constant. Thus, constant zero values present in the Dalvik bytecode have ambiguous typing that must be recovered.

*Comparison of object references*. The Java bytecode uses typed opcodes for the comparison of object references (if acmpeq and if acmpne) and for null comparison of object references (ifnull and ifnonnull).

*Storage of primitive types in arrays*. The Dalvik bytecode uses ambiguous opcodes to store and retrieve elements in arrays of primitive types (e.g., aget for int/ float and aget-wide for long/double) whereas the corresponding Java bytecode is unambiguous. The array type must be recovered for correct translation.

The ded decompiler

Building a decompiler from DEX to Java for the studyproved to be surprisingly challenging. On the one hand, Java decompilation has been studied since the 1990s— tools such as Mocha [5] date back over a decade, with many other techniques being developed [39, 32, 31, 4,3, 1]. Unfortunately, prior to our work, there existed no functional tool for the Dalvik bytecode.1 Because of the vast differences between JVM and DVM, simple modification of existing decompilers was not possible. This choice to decompile the Java source rather than operate on the DEX opcodes directly was grounded in two reasons. First, we wanted to leverage existing tools for code analysis. Second, we required access to source code to identify false-positives resulting from automated code analysis, e.g., perform manual confirmation. Ded extraction occurs in three stages: *a*) retargeting,*b*) optimization, and *c*) decompilation.

## III. INTRODUCTION PENETRATION TESTING

A penetration test is a method of evaluating the security of a computer system or network by simulating anattack from a malicious source, known as a Black Hat Hacker, or Cracker. The process involves an active analysis of the system for any potential vulnerabilities that may result from poor or improper system configuration, known and/or unknown hardware or software flaws, or operational weaknesses in process or technical countermeasures. This analysis is carried out from the position of a potential attacker, and can involve active exploitation of security vulnerabilities. Any security issues that are found will be presented to the system owner together with an assessment of their impact and often with a proposal for mitigation or a technical solution. The intent of a penetration test is to determine feasibility of an attack and the amount of business impact of a successful exploit, if discovered. It is a component of a full security audit[1]. Penetration testing is one of the oldest methods for assessing the security of a computer system. In the early 1970's, the Department of Defense used this method to demonstrate the security weaknesses in computer systems and to initiate the development of programs to create more secure systems. Penetration testing is increasingly used by organizations to assure the security of Information systems and services, so that security weaknesses can be fixed before they get exposed[2]. But when the Penetration test is performed without a well-planned and professional approach – it can result to what it is supposed to prevent from. Penetration tests are typically aimed at environments prevalent in most organization, including Internet,intranet, extranet, and dial-up. While there are specific techniques for each environment, it is important to

-time consuming (e.g. a lot of time will be spent to re-order your test to “being-end” format)

- waste of effort (e.g. the testers might end up testing the same thing)

ISSN: 2321-9653



- ineffective testing (e.g. the results and the reporting might not suit the requirements of the client)

Methodology is a “map” using which you will reach your final destination (end of test) and without a

methodology the testers might get “lost” (reach the abovementioned results)..

## IV. EFFECTIVE PENETRATION TEST

The idea behind penetration testing methodologies is that the penetration tester should follow a pre-scripted format during test as dictated by the methodology. The 3 popular ones that come to mind:

▪The OSSTMM

▪NIST 4-Stage Pen-Testing Guideline

▪ISSAF

The Open Source Security Testing Methodology Manual is a peer-reviewed methodology for performing

security tests and metrics. The OSSTMM test cases are divided into five channels which collectively test:

information and data controls, personnel security awareness levels, fraud and social engineering control levels,computer and telecommunications networks, wireless devices, mobile devices, physical security access controls, security processes, and physical locations such as buildings, perimeters, and military bases. The OSSTMM focuses on the technical details of exactly which items need to be tested, what to do before, during, and after a security test, and how to measure the results. OSSTMM is also known for its Rules of Engagement which define for both the tester and the client how the test needs to properly run starting from denying false advertising from testers to how the client can expect to receive the report. New tests for international best practices, laws, regulations, and ethical concerns are regularly added and updated.The National Institute of Standards and Technology (NIST) discusses penetration testing in SP800-115[1-2]. NIST's methodology is less comprehensive than the OSSTMM; however, it is more likely to be accepted by regulatory agencies. For this reason NIST refers to the OSSTMM.

The Information Systems Security Assessment Framework (ISSAF) is a peer reviewed structured frameworkfrom the Open Information Systems Security Group that categorizes information system security assessment into various domains and details specific evaluation or testing criteria for each of these domains. It aims to provide field inputs on security assessment that reflect real life scenarios. The ISSAF should primarily be used to fulfill an organization's security assessment requirements and may additionally be used as a reference for meeting other information security needs. It includes the crucial facet of security processes and, their assessment and hardening to get a complete picture of the vulnerabilities that might exist. The ISSAF however is still in its infancy[1].Any penetration testing methodology, no matter how well thought out, has limited usefulness. Because the goal behind penetration testing is to try to find as many serious vulnerabilities as possible. In order to do this, the "mindset" of an attacker must be developed. The assessed system or application must be viewed in all of the possible ways that it could be misused, abused and exploited[7]. Based on specific objectives to be achieved, the different penetration testing strategies include:

- External testing strategy. External testing refers to attacks on the organization's network perimeter using

procedures performed from outside the organization's systems, that is, from the Internet or Extranet. This test may be performed with non-or full disclosure of the environment in question. The test typically begins with publicly accessible information about the client, followed by network enumeration, targeting the company's externally visible servers or devices, such as the domain name server (DNS), e-mail server,Web server or firewall.

•Internal testing strategy. Internal testing is performed from within the organization's technology environment. This test mimics an attack on the internal network by a disgruntled employee or an authorized visitor having standard access privileges. The focus is to understand what could happen if the network perimeter were successfully penetrated or what an authorized user could do to penetrate specific information resources within the organization's network. The techniques employed are similar in both types of testing although the results can vary greatly.

•Blind testing strategy. A blind testing strategy aims at simulating the actions and procedures of a real

ISSN: 2321-9653



hacker. Just like a real hacking attempt, the testing team is provided with only limited or no information concerning the organization, prior to conducting the test. The penetration testing team uses publicly

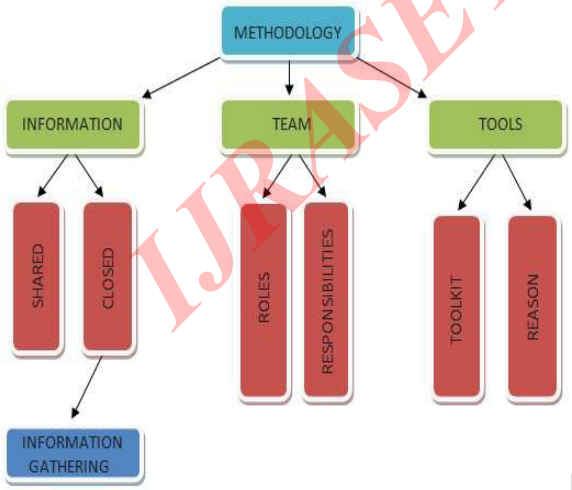
available information (such as corporate Web site, domain name registry, Internet discussion board,

USENET and other places of information) to gather information about the target and conduct its

penetration tests. Though blind testing can provide a lot of information about the organization (so called inside information) that may have been otherwise unknown, for example, a blind penetration may uncover such issues as additional Internet access points, directly connected networks, publicly available confidential/proprietary information, etc. But it is more time consuming and expensive because of the effort required by the testing team to research the target.

•Double blind testing strategy. A double-blind test is an extension of the blind testing strategy. In this exercise, the organization's IT and security staff are not notified or informed beforehand and are "blind" to the planned testing activities. Double-blind testing is an important component of testing, as it can test the organization's security monitoring and incident identification, escalation and response procedures. As clear from the objective of this test, only a few people within the organization are made aware of the testing.

Proposed Methodology




[Fig. 1]

Penetration testing methodology

While there are several available methodologies for you to choose from, each penetration tester must have their own methodology planned and ready for most effectiveness and to present to the client. In the prosposed methodology planning, there are 3 main figures that must be fully understood and followed:

1. Information. There are commonly 2 types of penetration testing:

•When the information about the organization is Closed (Black box) - the pen-tester performs the attack

with no prior knowledge of the infrastructure, defence mechanisms and communication channels of the

target organization. Black box test is a simulation of an unsystematic attack by weekend or wannabe

hackers (script kiddies).

•And when the information is Shared (White box) - the pen-tester performs the attack with full knowledge

of the infrastructure, defence mechanisms and communication channels of the target organization. White

About Effective Penetration Testing Methodology 430 box test is a simulation of a systematic attack by well prepared outside attackers with insider contacts or insiders with largely unlimited access and privileges.

If the penetration testers are using the “Black Box” approach, then Information gathering must be planned

out, because information gathering is one of the most important processes in penetration testing and it’s one of first phases in security assessment and is focused on collecting as much information as possible about a target application. This task can be carried out in many different ways: by using public tools (search engines), scanners, sending simple HTTP requests, or specially crafted requests, it is possible to force the application to leak information, e.g., disclosing error messages or revealing the versions and technologies used.[9] If the penetration testers are using the “White Box” approach, then the tester should target the information gathering procedure based on the scope (e.g. the clinet might give all the

required information, and might not want the testers to search for other information)

2. Team. Penetration testing is most effective if it's a team of professional, which all have their roles and

responsibilities appointed and all know what he/she must do and how to do it. In penetration testing, as in any sphere, each team member must know his/her part of the team, and should follow the affixed procedure (e.g. network administrator, should not be searching for vulnerabilities through the web-site) in order for the test to be quick, efficient and less time consuming. (e.g. security consultant is responsible to make the report clear and understandable, in order for the technicians to be more focused on testing rather than reporting)

3. Tools. And the last most important part of the test is the toolkit. Each penetration testers have their

"toolset" to perform a penetration test. These tools are usually chosen in order to make their work most

effective (a test cannot be effective if the owner of the system assigns tools, which the testers are not familiar with). There are many tools available, and many of them are available for free usage, but the penetration testers must have excellent usage at least with some of them, rather that know most of them on an average level. It is also vital for the testers to choose their toolkits wisely, since this not only one area to perform a penetration test in (software development, network). For example, network vulnerability scanners that try to evade detection by IDS and IPS devices would normally not be useful for software development. So the testers should choose the toolkit with features that are suitable for them (e.g. Configurability, Extensibility).

## CONCLUSION

One of the crucial factors in the success of a pen-test is the underlying methodology. Lack of a formal

methodology means no consistency, and the client wouldn't want to be paying and watching the testers

testing cluelessly. While a penetration tester's skills need to be specialized for the job, the approach shouldn't be. In other words, a formal methodology should provide a disciplined framework for conducting a complete and accurate penetration test. There are available methodologies that are very effective and already used by many penetration testers, some of which were given in the research, but authors point of view to the mothodology was given in the research.

## REFERENCES

[1] en.wikipedia.org/wiki/Penetration_testing

[2]http://searchnetworking.techtarget.com/generic/0,295582,sid7_gci1083683,00.html

[3] Kurtz, George and Chris Prosise, Penetration Testing Exposed - Part 3 'Audits, Assessments & Tests(Oh, My)', September, 2000

[4]http://www.corecom.com/external/livesecurity/pentest.html

[5] http://www.network-defense.com/papers/pentest.html

[6] Internet Security Systems, Network and Host-based Vulnerability Assessment

[7]http://www.infosecinstitute.com/blog/ethicalhacking_computer_forensics.html

[8]http://www.owasp.org/index.php/Testing:_Information_Gathering

[9] Fernflower - java decompiler. http://www.reversed-java.com/fernflower/.

[10] Fortify 360 Source Code Analyzer (SCA). https://www.fortify.com/products/fortify360/source-code-analyzer.html.

[11] Jad. http://www.kpdus.com/jad.html.

[12] Jd java decompiler. http://java.decompiler.free.fr/.

[13] Mocha, the java decompiler. http://www.brouhaha.com/~eric/software/mocha/.

[14] ADMOB. AdMob Android SDK: Installation Instructions.http://www.admob.com/docs/AdMob_Android_SDK_Instructions.pdf. Accessed November 2010.