VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

Fall SEMESTER 2022-23

Information Security Analysis and Audit

REVIEW-1

Identifying and fixing vulnerabilities in mobile Online Application (MOBSF)

Team Members

Abhist Chauhan – 19MIS0253

Ashok Nirmal – 19MIS0254

Komal Bhagat – 19MIS0380

Lauryn Arora- 20MIS0193

Yash Heda – 20MIS0300

Name : LAURYN ARORA
Register No.: 20MIS0193

Secure Mobile Software Development with Vulnerability Detectors in Static Code Analysis

ABSTRACT:
As we are observing the mobile devices being used for a wide variety of purposes the mobile applications that provide these products and services have become the major target of the malicious hackers. It has been suggested to identify and fix the security loopholes before an attack is made and confidential information is lost.

METHODOLOGY:

- A static security analysis approach has been employed that uses open source FindSecurityBugs plugin for Android Studio IDE.
    - Can be used to find security issues in java code and scan java, android and Scala applications.
    - Analysis is made at the bytecode level, availability of source code is not essential.
    - Feature allows the developer to design custom security vulnerability detectors.
    - For example, the built detectors can recognize a vulnerability of SQL injection and data leakage in an Android mobile application program, which may face the threat of potential malicious code injection, and then issue a warning on the code line.
- Categorize the common mobile vulnerability for developers based on OWASP mobile security recommendations.

RESULT:
It is very important that all of these vulnerabilities are reported accurately, there are two possible cases with SCA, that it either fails to report and warn about a true flaw or sends a false alarm about a fake flaw.

Name : LAURYN ARORA
Register No.: 20MIS0193

Analysis and Impact of Vulnerability Assessment and Penetration Testing

ABSTRACT:
The improvement in hardware and corresponding necessary improvements in software i.e. mobile applications although have improved the speed, efficiency and the user experience have resulted in highly complex systems with a greater scope of vulnerabilities. Therefore, it is better to recognize these before an attack takes place. This paper discusses VAPT process and VAPT tools for finding vulnerabilities in the system.

METHODOLOGY:

Vulnerability Assessment Testing Methods:

1. Active Testing – In this process, new test data is introduced.
2. Passive Testing – In this process, we just monitor the existing data, new test data is not introduced
3. Network Testing – In this process, we measure the current state of the network.

PENETRATION TECHNIQUES

1. *Functional Testing:*

Also known as black box testing, it is used to check the functionality of the system. It relies n passing varied inputs to the system and observing the corresponding outputs. No knowledge of the programming language is required.

2. *Grey Box Testing:*

It is the fusion of black box and white box testing. The testing engineer has partial information or refers to the source code of the program to be tested.

3. *Glass Box Testing:*

Also known as white box testing, it is used complementary to black box testing. But unlike black box, logical checking of source code along with path execution is done. It is performed by software developers and used for unit testing. Proficiency in a programming language s must.

RESULT:

This Android Studio FindSecurityBugs plugin can be used by people with ranging levels of technical knowledge and enable them to develop custom Android App vulnerability detectors and enhance the security of the application.

Name : LAURYN ARORA
Register No.: 20MIS0193

Secure Mobile IPC Software Development with Vulnerability Detectors in Android Studio

ABSTRACT:
The shift of the audience to using smartphones for almost every purpose, has also attracted attention of hackers and security attacks on mobile applications are on the rise. Faults, bugs and errors in apps ,especially relating to the passing of intent are loopholes from where the malicious users can creep in and compromise the crucial data and functionalities of the system. Timely detection and elimination of these security holes will alleviate the security risk threats and strengthen the security of the application.

METHODOLOGY:

A. FindSecurityBugsDetectors for Secure Android Software Development
The developed plugin would protect the application against security risks such as SQL injection, data leakage and insecure data storage. It works by first parsing the Android java source code, recognises the specific API calls and warns about the probable risks. It also recommends solutions and suggestions to replace code lines.

B. Vulnerable Broadcast Intent Detectors
Communication between different application components via intent can be a source major security risks but such risks can be mitigated by using intent vulnerability detector that alerts the developer and also recommends security measures that can be taken.

RESULT:
In this paper, an IPC(Inter Process Communication) flaw detector build using FindSecurityBugs plugin for Secure Mobile Software Development(SMSD) is discussed. The built package can be loaded into the Android Studio IDE and is ready to use. It will help to improve SMSD knowledge as well as skills among the android application developers.

YASH HEDA
20MIS0300

Neutralizing vulnerabilities in Android: a process and an experience report

Abstract:

**Mobile devices became a natural target of securitythreats due their vast popularization. That problem is even moresevere
when considering Android platform, the market leaderoperating system, built to be open and extensible. Althoug hAndroid provides security countermeasures to handle mobilethreats, these defense measures are not sufficient and attacks canbe performed in this platform, exploiting existing
vulnerabilities.Then, this paper focuses on improving the security of theAndroid ecosystem with a contribution that is two-fold, as follows:i) a process to analyze and mitigate Android vulnerabilities,scrutinizing existing security breaches found in the literature andproposing mitigation actions to fix them; and ii) an experiencereport that describes four vulnerabilities and their corrections,being one of them a new detected and mitigated vulnerability.**

Methodology:

i) a process to analyze and mitigateAndroid vulnerabilities, scrutinizing existing security
breachesfound in the literature and proposing mitigation actions tofix them; and
ii) an experience report that describes fourvulnerabilities and their correction, being one of them a newdetected and mitigated vulnerability

conclusion:

In this paper, we reported our experience in the analysisand mitigation of Android vulnerabilities. The high number of existing vulnerabilities and the impact of them in the Androidsecurity show the importance of this topic and the necessityof devising defense mechanisms.

YASH HEDA
20MIS0300

Mobile Applications -Vulnerability Assessment Through the Static and Dynamic Analysis

Abstract :

In the recent day's mobile applications usage is increasing by banking and financial institutes, health and hospital management systems such as mobile banking apps, e-commerce apps, news feeds, inpatient and outpatient information, social networking apps and game apps etc. All these mobile applications require support for security properties like authentication, authorization, data confidentiality, and sensitive information leakage etc. The mobile applications have seen rapidly growth in the last couple of years. These applications has provided suitable to banks, credit card data, personally identified information, travel applications etc., the enterprise mobile applications extend corporate networks beyond the perimeter devices and thus potentially expose these organizations to the new types of security threats. Security risks associated with these applications can often be identified and mitigated by subjecting them to security testing. Compared to desktop or web applications, mobile applications are hard to test for security. At the same time, these applications are not necessarily more secure then desktop or web applications.

Methodology:

In this paper, we discuss the importance of mobile applicationsvulnerability assessment and penetration testing techniques to identify different mobile application threats and evaluate our contribution on the different flavors of mobile development platforms such as iOS for iPAD and iPhone apps, Android apps, Windows apps and BlackBerry applications.

Conclusion:

Mobile application developers and the organizations intending developing and deploy mobile applications in secure manner and must plan their security testing called penetration testing and they follow strategy across manual and automation tests approaches for efficient and error-free application delivery. In addition to actual devicebased testing, emulators should be included as an integral part of the security testing program. Enterprise applications require special pen testing techniques because they deal with lot users sensitive data. Outsourcing to vendors who are operating an independent testing practice may be a viable option to manage the expertise, scalability, and quality assurance requirements for mobile application delivery in secure manner.

YASH HEDA
20MIS0300

Vulnerability Detection on Android Apps–Inspired by Case Study on Vulnerability Related with Web Functions

ABSTRACT:

Nowadays, people's lifestyle is more and more dependent on mobile applications (Apps), such as shopping, financial management and surfing the internet. However, developers mainly focus on the implementation of Apps and the improvement of user experience while ignoring security issues. In this paper, we perform the comprehensive study on vulnerabilities caused by misuse of APIs and form a methodology for this type of vulnerability analysis. We investigate the security of three types of Android Apps including finance, shopping and browser which are closely related to human life. And we analyze four vulnerabilities including Improper certificate validation, WebView bypass certificate validation vulnerability, WebView remote code execution vulnerability and Alibaba Cloud OSS credential disclosure vulnerability. In order to verify the effectiveness of our analysis method in large-scale Apps on the Internet, we propose a novel scalable tool - VulArcher, which is based on heuristic method and used to discover if the above vulnerabilities exist in Apps. We download a total of 6114 of the above three types of samples in App stores, and we use VulArcher to perform the above vulnerability detection for each App. We perform manual verification by randomly selecting 100 samples of each vulnerability. We find that the accuracy rate for ACOCDV can reach 100%, the accuracy rate for WBCVV can reach 95%, and the accuracy rate for the other two vulnerabilities can reach 87%. And one of vulnerabilities detected by VulArcher has been included in China National Vulnerability Database (CNVD) ID(CNVD-2017-23282). Experiments show that our tool is feasible and effective.

Methodology:

y. In order to verify the effectiveness of our analysis methodology in large-scale Apps on the Internet. Based on our findings, we propose a vulnerability detection tool, named VulArcher. It can detect the below four vulnerabilities in packed or unpacked Android Apps,and the average accuracy rate is 91%.

WRCEV, WBCVV, ICV and ACOCDV. So we chose these four vulnerabilities as our study goals

Conclusion:

We deeply analyzed the four kinds of vulnerabilities on a data set with more than 6000 Android Apps. The aforementioned vulnerabilities are ACOCDV, WRCEV, WBCVV and ICV. We found that webview and HTTPs are existed in most of finance and shopping apps. According to this, attackers can easily perform MITM. As for WRCEV which has been disclosed for a long time, it still exist in browser Apps, because developers lack the methods of vulnerability identification and security development awareness. Apps that use Alibaba Cloud OSS services, information leakage vulnerability of Alibaba cloud OSS credentials is caused by the weak secure awareness of developers. We developed a vulnerability detection tool, VulArcher. The experiments show that it can automatically detect the above vulnerabilities, and has good scalability. One of vulnerabilities which VulArcher detected had been included in China National Vulnerability Database (CNVD) ID(CNVD-2017-23282). It detected more than 6000 Apps and found nearly 3000 Apps with the above vulnerabilities. and we have manually verified the accuracy of results among nearly 300 Apps. The source codes of VulArcher and sample data presented in this paper can be utilized by further researchers. We incentivize this by making our data publicly available

Paper-1 **Komal Bhagat**
19MIS0380

Patchworking: Exploring the code modifications brought on by vulnerability-fixing actions

Abstract:

Identifying and repairing vulnerable code is a critical software maintenance task. The ability to assess the potential impacts of a change before implementing it makes change impact analysis a crucial component of software maintenance. There are no methods for predicting the impact when a change entails fixing a security flaw, despite the fact that the software engineering community has extensively examined techniques and tools for performing impact analysis of change requests.

Methodology:

In order to test our hypothesis, we look at 524 security patches that were issued to 98 distinct Java open-source projects' vulnerabilities that fell into 10 different weakness categories.

Result:

On the one hand, our discoveries pave the path for better software maintenance activity management when addressing software vulnerabilities. In fact, vulnerability class information could be used to more accurately anticipate how much code will be fixed, how the structural characteristics of the code (such as complexity, coupling, cohesiveness, and size) will change, and how much work will be involved in correcting it. On the other side, our findings can be used to enhance automated procedures that aid engineers in addressing security problems.

Paper – 2

Komal Bhagat
19MIS0380

Investigating the vulnerability fixing process in OSS projects: Peculiarities and challenges

Abstract:

Vulnerabilities can be thought of as bugs and are treated as such, but they differ from other bugs in many ways (canonical bugs in the remainder of the paper). A vulnerability increases a system's functionality by allowing an attacker to abuse or misuse it, whereas a canonical defect results from the wrong or inadequate implementation of a requirement, which reduces the system's functionality. This variation may have an impact on how vulnerabilities are fixed.

Methodology:

From the aforementioned goal, we can confirm It takes more re-assignments (than those seen in canonical bugs) to identify developers who can deal with bugs connected to vulnerabilities. The efficiency of security bug assignment duties should be improved, which will reduce the number of re-assignments. Vulnerabilities take less time to fix than canonical defects but require more effort, contributors, and time to establish the fixing strategy.

Result:

The paper presents a case study investigating the use of (modified) vulnerable OSS and their impact on vulnerability scanners. We found that such modifications heavily decrease the precision and recall of vulnerability scanners by checking the performance of the open-source scanners DependencyCheck and Eclipse Steady, GitHub Security Alerts as well as three commercial tools. The results show that all vulnerability scanners struggle to cope with modified OSS, and that further research and development are needed.

Paper – 3

Komal Bhagat
19MIS0380

A Manually-Curated Dataset of Fixes to Vulnerabilities of Open-Source Software

Abstract:

Advancing our understanding of software vulnerabilities, automating their identification, the analysis of their impact, and ultimately their mitigation is necessary to enable the development of software that is more secure. While operating a vulnerability assessment tool, which we developed, and that is currently used by hundreds of development units at SAP, we manually collected and curated a dataset of vulnerabilities of open-source software, and the commits fixing them. The data were obtained both from the National Vulnerability Database (NVD), and from project-specific web resources, which we monitor on a continuous basis.

Methodology:

The availability of mature, high-quality open-source software (OSS) components represents an opportunity for the software industry to accelerate innovation and lower costs, but, at the same time, maintaining a secure OSS supply chain and an effective vulnerability management process is a significant challenge.

Result:

We have presented a dataset of fixes to vulnerabilities in Java OSS projects of industrial relevance, resulting from our experience with developing and operating an open-source vulnerability management solution at SAP. While the data we are releasing at this time cover only Java projects, in the meantime the tool has evolved to support more languages and consequently we are working to extend our dataset further.

ABHIST CHAUHAN
19MIS0253

1.Security analysis of third-party in-app payment in mobile applications.

Abstract:

The massive growth of smart mobile devices has attracted numerous applications to embed third-party in-app payments, which involves more sophisticated interactions between multiple participants compared to traditional payments. studied four common third-party mobile payment cash registers and summarized the security rules that cashiers, and merchants must regulate. they also discovered seven cases of security breaches on both the Android and iOS platforms.

METHODOLOGY:

Feature based identification strategy- to detect apps with TP-SDK and reverse-engineer TP-SDKs of four cashiers on two platforms and extract their unique features.

Types of attacks performed:

1. Order Tampering
2. notification forging attack
3. Unauthorized querying
4. Order substituting

RESULT:

Insecure in-app payment is becoming a main threat to mobile ecosystem as more and more online transactions are transferring from website to app

Their analysis investigates implementations of four in-app payments and concludes a series of security rules that should be obeyed and apps integrated with third-party in-app payment SDKs are also vulnerable.

ABHIST CHAUHAN
19MIS0253

2.An Empirical Study on Android-Related Vulnerabilities.

Abstract:

Mobile devices are increasingly used in everyday life. In this article, the authors present the study to date investigating Android-related vulnerabilities, with a special focus on those affecting the Android OS.

For example, (i) define a detailed taxonomy of Android-related vulnerability types, (ii) examine the layers and subsystems of the Android operating system affected by the vulnerability, and (iii) study the survivability of the vulnerability.

METHODOLOGY:

1. Data Extraction and Analysis
2. Check and Complement the Vulnerability Type Automatically Inferred by CVE Details, and Obtain its Hierarchy.
3. Identify the Subsystems Affected by the Vulnerability.

For comparing the distributions of the survivability of the different categories of vulnerabilities.

(i) forest plots, and (ii) statistical tests

RESULT:

The achieved results show that most of the vulnerabilities are related to improper restriction of operations in the bounds of memory buffers, issues processing data, improper access control, and improper input validations.

Findings also indicate that third-party hardware drivers are the components mostly affected by security vulnerabilities in the Android OS, thus suggesting the strengthening of verification & validation activities performed on them.

ABHIST CHAUHAN
19MIS0253

3.Identifying vulnerabilities of SSL/TLS certificate verification in Android apps with static and dynamic analysis

Abstract:

Many Android developers fail to properly implement SSL/TLS during application development, which can lead to Man-In-The-Middle (MITM) or phishing attacks. Design and implement of a tool called DCDroid to detect vulnerabilities using a combination of static and dynamic analysis is done.

METHODOLOGY:

DCDroid (Detecting vulnerable Certificates in Android apps)-

1.In static detection, vulnerable code is defined

2.In dynamic detection, the apps are run under the guide of static analysis. -An activity with vulnerable code does not start directly so that the tool is more stable in the detection

3. Next proxy servers are set up to carry out MITM attacks.

4. Development of an app to capture traffic on the smart phone so that we can get the pure traffic of apps and reduce the false positives.

RESULT:

In this work, the authors proposed an effective method and developed a tool called "DCDroid" to identify vulnerabilities in the implementation of SSL/TLS digital certificate verification in Android. they analyzed 960 apps from Google Play and 1253 apps from 360app. Extensive experimental results show that based on initial static analysis, 457 (20.65%) applications contain potential security risks when implementing SSL/TLS.

19MIS0254
ASHOK NIRMAL

An Exploratory Analysis of Mobile Security Tools

Abstract:
The growing market of the mobile application is overtaking the web application. Mobile application development environment is open source, which attracts new inexperienced developers to gain hands on experience with applicationn development. However, the security of data and vulnerable coding practice is an issue. Among all mobile Operating systems such as, iOS (by Apple), Android (by Google) and Blackberry (RIM), Android dominates the market. The majority of malicious mobile attacks take advantage of vulnerabilities in mobile applications, such as sensitive data leakage via the inadvertent or side channel, unsecured sensitive data storage, data transition and many others. Most of these vulnerabilities can be detected during mobile application analysis phase. In this paper, we explore vulnerability detection for static and dynamic analysis tools. We also suggest limitations of the tools and future directions such as the development of new plugins.

Security Analysis of Mobile Banking Application

Abstract:
This paper discusses the security posture of Android m-banking applications in Qatar. Since technology has developed over the years and more security methods are provided, banking is now

heavily reliant on mobile applications for prompt service delivery to clients, thus enabling a seamless and remote transaction. However, such mobile banking applications have access to sensitive data for each bank customer which presents a potential attack vector for clients, and the banks. The banks, therefore, have the responsibility to protect the information of the client by providing a high-security layer to their mobile application. This research discusses m-banking applications for Android OS, its security, vulnerability, threats, and solutions. Two m-banking applications were analyzed and benchmarked against standardized best practices, using the combination of two mobile testing frameworks. The security weaknesses observed during the experimental evaluation suggest the need for a more robust security evaluation of a mobile banking application in the state of Qatar. Such an approach would further ensure the confidence of the end-users. Consequently, understanding the security posture would provide a veritable measure towards mbanking security and user awareness.

Automated Android Applications Vulnerability Detection, a Hybrid Static and Dynamic Analysis Approach

Abstract:

The security of mobile applications has become a major research field which is associated with a lot of challenges. The high rate of developing mobile applications has resulted in less secure applications. This is due to what is called the “rush to release” as

defined by Ponemon Institute. Security testing—which is considered one of the main phases of the development life cycle—is either not performed or given minimal time; hence, there is a need for security testing automation. One of the techniques used is *Automated Vulnerability Detection*. Vulnerability detection is one of the security tests that aims at pinpointing potential security leaks. Fixing those leaks results in protecting smart-phones and tablet mobile device users against attacks. This paper focuses on building a hybrid approach of static and dynamic analysis for detecting the vulnerabilities of Android applications. This approach is capsuled in a usable platform (web application) to make it easy to use for both public users and professional developers. Static analysis, on one hand, performs code analysis. It does not require running the application to detect vulnerabilities. Dynamic analysis, on the other hand, detects the vulnerabilities that are dependent on the run-time behaviour of the application and cannot be detected using static analysis. The model is evaluated against different applications with different security vulnerabilities. Compared with other detection platforms, our model detects information leaks as well as insecure network requests alongside other commonly detected flaws that harm users' privacy. The code is available through a GitHub repository for public contribution.

Refrences:

1. X. Meng, K. Qian, D. Lo, P. Bhattacharya and F. Wu, "Secure Mobile Software Development with Vulnerability Detectors in Static Code Analysis," 2018 International Symposium on Networks, Computers and Communications (ISNCC), 2018, pp. 1-4, doi: 10.1109/ISNCC.2018.8531071.

2. 2 . Y. Khera, D. Kumar, Sujay and N. Garg, "Analysis and Impact of Vulnerability Assessment and Penetration Testing," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), 2019, pp. 525-530, doi: 10.1109/COMITCon.2019.8862224.

3. X. Meng, K. Qian, D. Lo, H. Shahriar, M. D. A. I. Talukder and P. Bhattacharya, "Secure Mobile IPC Software Development with Vulnerability Detectors in Android Studio," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 2018, pp. 829-830, doi: 10.1109/COMPSAC.2018.00141.

4. Ponta, Serena Elisa, et al. "A manually-curated dataset of fixes to vulnerabilities of open-source software." 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR). IEEE, 2019.

5. Canfora, Gerardo, et al. "Investigating the vulnerability fixing process in oss projects: Peculiarities and challenges." Computers & Security 99 (2020): 102067.

6. Canfora, Gerardo, et al. "Patchworking: Exploring the code changes induced by vulnerability fixing activities." Information and Software Technology 142 (2022): 106745.

7. Yang, Wenbo, et al. "Security analysis of third-party in-app payment in mobile applications." Journal of Information Security and Applications 48 (2019): 102358.

8. Linares-Vásquez, Mario, Gabriele Bavota, and Camilo Escobar-Velásquez. "An empirical study on android-related vulnerabilities." 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR). IEEE, 2017.

9. Wang, Yingjie, et al. "Identifying vulnerabilities of SSL/TLS certificate verification in Android apps with static and dynamic analysis." Journal of Systems and Software 167 (2020): 110609.

10. Shahriar, Hossain, Md Arabin Talukder, and Md Saiful Islam. "An exploratory analysis of mobile security tools." (2019).

11. Al-Delayel, S. A. (2022). Security Analysis of Mobile Banking Application in Qatar. arXiv preprint arXiv:2202.00582.

12. Amin, Amr, et al. "Androshield: Automated android applications vulnerability detection, a hybrid static and dynamic analysis approach." Information 10.10 (2019): 326.

13. de Carvalho, Carlos André Batista, et al. "Neutralizing vulnerabilities in android: A process and an experience report." International Journal of Computer Science and Information Security 14.3 (2016): 20.
14. Basavala, Sreenivasa Rao, Narendra Kumar, and Alok Aggarwal. "Mobile applications-vulnerability assessment through the static and dynamic analysis." Conference on Advances in Communication and Control Systems (CAC2S 2013). Atlantis Press, 2013.
15. Qin, Jiawei, et al. "Vulnerability Detection on Android Apps–Inspired by Case Study on Vulnerability Related With Web Functions." IEEE Access 8 (2020): 106437-106451.

Base Paper:

We have selected **"Analysis and Impact of Vulnerability Assessment and Penetration Testing"** as our base paper as it discusses about a wide arena of testing tools (like Wireshark, Nmap, Metasploit) and techniques that can be employed to test our online mobile applications for possible risks and security breaches. The most common types of security attacks and the ways to mitigate them have also been discussed. The detailed life cycle of penetration testing and vulnerability assessment has been explained. It also talks about functional and network testing for tracking vulnerabilities in the android application. In a nutshell, most of the relevant technologies relating to android security have been discussed which would provide us with guidelines to implement our project.