

## **Security Assessment for Digital Wallet Payment Partner Applications using the OWASP Method: A Case Study in Indonesia**

Muhammad Saifulhakim, Ahmad Nurul Fajar

Information System Management Department, BINUS Graduate Program – Master of Information System Management, System Management, Bina Nusantara University, Jakarta, Indonesia 11480

*muhammad.saifulhakim@binus.ac.id, afajar@binus.edu*

**Abstract.** This study presents a security assessment of a digital wallet payment partner mobile application using the Open Web Application Security Project (OWASP) framework. A qualitative method approach for data retrieval involves conducting interviews and observing test results following the OWASP Mobile Application Security Testing Guide (MASTG) and the OWASP Mobile Application Security Verification Standard (MASVS) verification standard. Based on observations from penetration testing using both Static (SAST) and dynamic (DAST) analysis methods, six vulnerabilities were identified out of 84 MASVS test cases, with 10 being deemed not applicable. Vulnerabilities were evaluated using the OWASP Risk Rating Methodology, resulting in a moderate likelihood rating of 5.01 and a moderate impact rating of 4.26. The overall risk matrix rating is assessed as moderate. The proposed solution aims to mitigate the identified vulnerabilities and enhance application security against potential attacks. This study showcases the effectiveness of the OWASP methodology for conducting a comprehensive assessment of mobile application security risks.

**Keywords:** Security assessment, OWASP, OWASP Risk Rating Methodology, Information Security

## **1. Introduction**

In the current era of rapid technological development, the extensive use of smartphones in daily activities is closely tied to numerous applications in circulation. Mobile applications have become integral to everyday life, serving functions from communication and entertainment to transactions (Ali et al., 2019). However, amidst the convenience and advantages of these applications, particularly mobile electronic money apps, there are considerations to be addressed, notably concerning security. Security has become a paramount concern for developers and researchers, aiming to identify vulnerabilities, assess their impact, and devise ways to safeguard against these weaknesses (Mei-Ling et al, 2018). Within the realm of information system security, three core aspects: confidentiality, integrity, and availability must be considered when implementing security measures (Björn & Niklas, 2017). These aspects are vital for establishing security in information systems, including mobile applications.

Currently, mobile devices are frequently targeted by hackers. This is due to the presence of human or technical errors during the development of mobile applications, which introduce gaps and vulnerabilities in the application's structure. These vulnerabilities offer opportunities for attackers to engage in cyber-attacks. Several articles and studies highlight instances of attacks and the discovery of vulnerabilities within mobile applications. According to a report by Atlas VPN, citing data from the Synopsys Cybersecurity Research Center (CyRC), 63% of Android applications exhibited security vulnerabilities in Q1 2021, averaging 39 vulnerabilities per application. Gaming and financial apps were particularly vulnerable, with financial transaction applications showing an 80% vulnerability rate (Ruth, 2021). Furthermore, data analysis reports from MediaIndonesia.com indicate that financial institutions in Indonesia face an average of 2,730 weekly attacks in the past six months (2022), surpassing the global average by 252% (mediaindonesia.com, 2022). Additionally, many organizations compromise their mobile app security levels. Guard Square's article, based on Verizon's 2020 mobile security index survey, states that 43% of companies sacrificed their cellular security, often opting for expedited marketing over comprehensive security testing (Guardsquare, 2020). To identify and mitigate vulnerabilities in mobile applications, conducting security risk assessments, particularly penetration testing, is essential. Among various methods such as ISSAF, OSSTMM, and NIST, the OWASP framework is well-suited for security risk assessment (Paul et al., 2021). OWASP (Open Web Application Security Project) provides an open-source framework to guide security assessments on mobile applications, which is well-documented through literature studies and research.

This is supported by several traced articles and research journals. In the findings of a report produced by a security development agency, Positive Technologies, it was revealed that Android-based mobile applications possess vulnerabilities with a high risk, reaching up to 43% (Positive Technologies, 2019). Among the vulnerabilities frequently encountered in mobile applications, insecure data storage stands out with a vulnerability rating of up to 76%. Statistical data concerning mobile application vulnerability risks, as discovered by Positive Technologies, indicates that insecure data storage ranks second in the OWASP Top 10 Mobile Risk 2016 category. Pradeo, a security development agency, asserted that their analysis unveiled that 74% of mobile applications distributed via Google Play were susceptible to at least one out of the top 10 risks outlined by OWASP (Pradeo, 2022). Multiple literature studies and articles conducted by researchers and cyber security service development organizations have consistently highlighted that OWASP presents a viable solution for security assessment. These findings underscore that OWASP serves as a valuable reference for conducting penetration testing and security assessments.

A security assessment will be carried out on digital wallet payment partner applications owned by digital wallet companies in Indonesia. These applications function as transaction service provider platforms connecting partners who collaborate, such as MSME partners, with the company's digital wallet application. They also serve as profile dashboards for partners associated with the company. Several factors underlie the need for a security assessment of the application. Here are some of the reasons:

1. Continuous Development: As development progresses, more features will be added, and elements will change frequently. Regular penetration testing and risk assessments need to be conducted to ensure that there are no gaps that could be exploited.
2. ISO 27001 Compliance: Another driving factor for the security risk assessment is the company's effort to comply with ISO 27001 requirements. These requirements are currently being implemented within the company, and the necessity for compliance (Robertus & Ditdit, 2023).

As a result, research was carried out to identify vulnerabilities within payment partner applications, with the goal of offering insights in the form of vulnerability assessment results and analyses using the OWASP method. Consequently, these findings can contribute to evaluation outcomes, propose enhancement strategies, and offer suggestions for potential improvements. This case study also involves the pursuit of research questions, which will serve as a foundation for framing the problem in this study. The subsequent list outlines several research questions that will be explored in this case study:

RQ1 : How can vulnerabilities be identified in payment partner applications?

RQ2 : What are the outcomes of data and vulnerability analysis of payment partner applications using OWASP?

RQ3 : What are the assessment findings and recommended mitigation strategies that should be implemented for payment partner applications?

This case study aims to provide a basis for decision-making in mitigating vulnerabilities and enhancing mobile app security for the digital wallet company. By evaluating and addressing existing security concerns, the research endeavors to contribute to the future development of more secure mobile applications and deter attacks on payment partner applications in Indonesia.

## **2. Literature Review**

### **2.1. Digital Wallet**

Currently, social life in Indonesia cannot be separated anymore by digital media. In this era, technology allows us to be able to make transactions anywhere and anytime just by using a smartphone and internet connectivity. This was realized because in Indonesia itself the emergence of various digital wallet applications that are spread on the Google Play Store. Digital wallet itself is part of a FinTech (Financial Technology) based company which is a digital payment tool that uses server-based electronic media. In general, digital wallets are in the form of server-based applications and require an internet connection or network (Handayani & Novitasari, 2020). Currently, in Indonesia itself there are many digital wallet applications circulating on the Google Play Store. According to data from Bank Indonesia, in Indonesia, there was a significant increase in the percentage of digital wallet usage during the COVID-19 pandemic in 2020, reaching 44% (Carla, 2023).

### **2.2. OWASP**

OWASP, also known as the Open Web Application Security Project, is an open community dedicated to improving security measures in many aspects. The aspects tested to measure a level of increased security are web and mobile applications (Anthony & Yohan, 2021). All OWASP tools, documents, and forums are open source for those who wish to learn about and improve the security of an application.

### **2.3. OWASP Risk Rating Methodology**

The OWASP Risk Rating Methodology is a simple approach to calculating and assessing the risk associated with an application. where with this method it can be decided what to do with these risks by calculating the factors that have been determined in according with OWASP. In the OWASP Risk Rating Methodology method there are 6 process steps that can be carried out. The 6 processes are:

#### **2.3.1. Identifying a Risk**

This is the first step in establishing the OWASP Risk Rating Method process. The process of identifying

security risks needs to be assessed and testers need to gather information about the threat agents involved, the attacks to be used, the vulnerabilities involved, and the impact successful exploits have on the business.

#### 2.3.2. Factor of Estimating Likelihood

This is the second step in the OWASP Risk Rating Method process. This step aims to be able to predict the probability of a successful attack from the attacker. This estimating process is carried out after the risk identification is completed.

#### 2.3.3. Factor of Estimating Impact

This is step 3 of the OWASP Risk Rating Method process. This step aims to consider the impact of a successful attack. OWASP revealed that there are 2 types of impact that must be considered, namely technical impact and business impact

#### 2.3.4. Determining Severity of Risk

This is step 4 of the OWASP Risk Rating Method process. This step aims to determine the level / level of severity of a risk. In determining the severity of a risk, OWASP provides a scale from 0 to 9 which is divided into 3 parts, namely:

- 0 to < 3 low impact
- 3 to < 6 moderate impacts
- 6 to 9 high impacts

#### 2.3.5. Deciding What to Fix

This is step 5 of the OWASP Risk Rating Method process. This step aims to be able to decide what must be repaired from the risks that have been found previously. After the risks to the application have been classified, there will be a priority list of what needs to be fixed. In OWASP the general rule of thumb is that the repair that should come first is the most severe risk. OWASP concluded that not all risks are worth repairing, and some losses are not only expected, but also justifiable based on the cost of fixing the problem.

#### 2.3.6. Customizing Your Risk Rating Model

This is an optional step in the OWASP Risk Rating Methodology process. This step aims to establish a customizable risk rating framework, which is crucial for adoption in a business context.

### 2.4. OWASP MASVS

OWASP implements a security verification standard known as the Mobile Application Security Verification Standard (MASVS). This standard aims to standardize these requirements by employing verification levels that are suitable for various threat scenarios (Francisco et al., 2019). MASVS can serve as a valuable guide when performing application security testing. The following is an overview, as per Carlos et al. (2022), outlining the levels of security verification.

### 2.5. OWASP MASTG (Mobile Application Security Testing Guide)

The OWASP MSTG is a testing guide method released by the non-profit organization OWASP. It serves as a comprehensive guide that assists penetration testing teams in effectively following these guidelines during their tasks (Carlos et al., 2022). OWASP MSTG and OWASP MASVS are closely connected in terms of verifying these security standards and conducting testing accordingly. In this study, the testing focus was placed on Android-based mobile applications.

### 2.6. Similar Research

In preparing this research report, the authors analyzed previous studies that had been conducted by other researchers in a similar field and had even served as references for preparing this research report. This research analysis aims to demonstrate the applicability of the OWASP method/framework for conducting security assessments. Several similar studies that have been conducted are described as

follows. The first study was conducted by Noppanat & Vasaka (2018) conducted a study aiming to investigate whether this application can expose sensitive data through communication channels. In their conducted experiments, they examined 3 hospital applications and 5 stock and trading applications using static and dynamic analysis techniques. From their findings, it was observed that each application had its own vulnerabilities that reflected OWASP risks. Ha et al. (2020) conducted research related to information security risk management in the e-Government system owned by the Vietnamese government. In their paper, they introduce a holistic approach to assessing information security risks based on qualitative and quantitative methods for Vietnam's e-Government system. They utilize the CVSS and OWASP assessment standards to measure information system risk. Kevin & Dennis (2019) conducted research to implement an authentication scheme using the Guillou-Quisquater protocol. In their paper, they subsequently performed a testing phase based on the Open Web Application Security Project (OWASP) penetration testing framework. Additionally, penetration testing was executed by an expert. Three potential vulnerabilities were identified, and risk estimates were calculated using the OWASP Risk Rating Methodology. Based on the OWASP risk rating, the research revealed that the identified vulnerabilities are of intermediate level. Lastly, Kai et al. (2019) carried out a study related to the analysis of security risks in mobile applications. They analyzed and discussed guidelines related to security requirements for application development, focusing on addressing threats, vulnerabilities, and weaknesses according to the OWASP Top 10 Mobile.

### **3. Methodology**

#### **3.1. Research Method**

In principle, this case study research is to carry out a security assessment of a payment partner's mobile application using a qualitative method approach and the OWASP method. John & J (2018), explained that qualitative research tends to collect data in the field at locations where participants experience problems or problems being studied, and collect data themselves through document inspection, observation, or interviewing participants. Robert (2018), in his book explains how the relationship between case study research and qualitative methods implicitly tends to assume that conducting case studies can be considered as an acceptable variant of conducting qualitative research. From the research conducted by Secil & Kwestan (2021), there are several advantages to conducting qualitative research, namely realizing a detailed description of a problem, raising research questions, knowing a group of participant perspectives, simplifying complex problems, and obtaining more subjective data and detail. Therefore, the data collection method in this study will be carried out using qualitative methods in the form of interviews and observations at the company. Then to search for security risks and their assessment will be carried out using the OWASP method. To be able to assess a security risk that exists in a mobile application, the application of the method can be done using the OWASP Risk Rating (Noppanat & Vasaka, 2018). According to them, explaining risk estimation in the OWASP methodology starts with the model:  $\text{Risk} = \text{Likelihood} * \text{Impact}$ .

#### **3.2. Theoretical Framework**

From several searches of study sources from literature and websites related to this research, a framework was created to become a reference in carrying out this research so that it becomes structured and systematic. The proposed framework will be presented in the following Fig. 1.

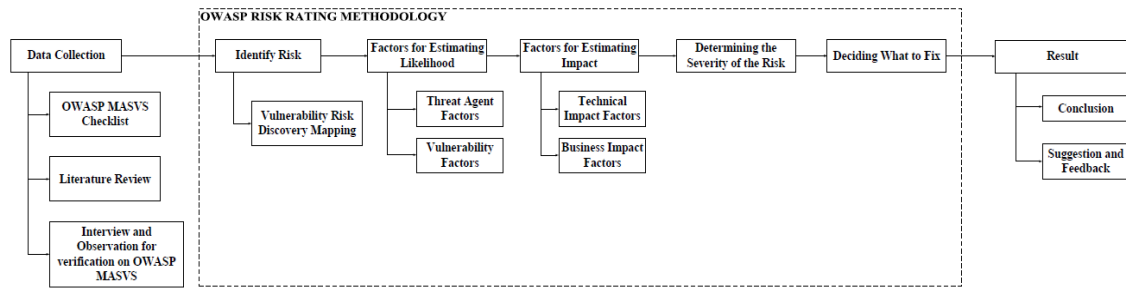


Fig. 1: Theoretical Framework

Broadly speaking, the framework in Fig. 1 consists of 7 stages of the work process. From each of the 7 main processes that will be carried out, there are several main processes that carry out a search for sub-processes. The 7 work processes to be carried out in this framework of thinking can be described in detail as follows:

### 3.2.1. Data Collection

This process involves data collection, which subsequently identifies the type of vulnerability and its associated risk level. There are three sub-processes involved in this data collection. The following details the specifics of these sub-processes:

- OWASP MASVS Checklist

The OWASP MASVS document as a standard for verifying data requirements to be used in the subsequent stage of the process. This document will serve as the foundation for the verification standard, which in turn becomes the reference point for the assessment. The OWASP MASVS version utilized is the latest one, currently version 1.5.0. Fig. 2 is a brief example of the OWASP MASVS checklist file, a checklist document that has been prepared obtained from the OWASP official website to be able to run payment partner mobile application testing at the digital wallet company.



Mobile Application Security

Verification Standard

OWASP MASTG v1.5.0 (commit: 3b9278f)

OWASP MASVS v1.4.2 (commit: 2a8b582)



OWASP

Mobile Security Project

Architecture, Design and Threat Modeling Requirements

| ID  | MASVS-ID    | Detailed Verification Requirement                                                                                                                     | L1 | L2 | R | Common | Android | Status    |
|-----|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|---|--------|---------|-----------|
| 1.1 | MSTG-ARCH-1 | All app components are identified and known to be needed.                                                                                             |    |    |   |        |         |           |
| 1.2 | MSTG-ARCH-2 | Security controls are never enforced only on the client side, but on the respective remote endpoints.                                                 |    |    |   |        |         | Test Case |
| 1.3 | MSTG-ARCH-3 | A high-level architecture for the mobile app and all connected remote services has been defined and security has been addressed in that architecture. |    |    |   |        |         |           |
| 1.4 | MSTG-ARCH-4 | Data considered sensitive in the context of the mobile app is clearly identified.                                                                     |    |    |   |        |         |           |
| 1.5 | MSTG-ARCH-5 | All app components are defined in terms of the business functions and/or security functions they provide.                                             |    |    |   |        |         |           |
| 1.6 | MSTG-ARCH-6 | A threat model for the mobile app and the associated remote services has been produced that identifies potential threats and countermeasures.         |    |    |   |        |         |           |

Fig. 2: Example of Document MASVS Checklist

In OWASP MASV Checklist there are 8 important requirements need to be tested. The 8 important requirements are: Architecture design and threat architecture, Data storage & privacy, Cryptography, Authentication & session management, Network communication, Platform interaction, Code quality and build setting, and Resilience.

- Literature Review

In this sub-process, the step to be taken involves conducting a literature review. The review will gather insights from various articles, similar journals, and websites of security development institutions. This study will elucidate the methods and approaches to be employed.

- Interview and Observation for Verification on OWASP MASVS

The second method of data collection involves qualitative techniques, particularly conducting interviews. In this study, interview data were gathered through discussions with three informants. The information source was personnel from the expert of Cyber Security IT team at a digital wallet company. The reason for involving three informants in the interview was that all security infrastructure and responsibilities related to application architecture. Furthermore, verification data collection was sought through observation. Observation encompassed the examination of application documents such as source code and business flow, as well as the results of penetration testing carried out at the company. This approach aims to comprehend how the application functions and analyze the outcomes of penetration testing conducted by the IT Cyber Security team for each identified point in the OWASP MASVS.

### 3.2.2. Identify Risk

This process involves identifying risks related to vulnerabilities obtained from the OWASP MASVS verification results. Within this process, there is one sub-process that will be executed, namely Vulnerability Risk Discovery Mapping. In this sub-process, a mapping of vulnerability risk discovery is carried out on OWASP MASVS points. This process aims to determine the value of each vulnerability risk.

### 3.2.3. Factors for Estimating Likelihood

his process is conducted to determine the likelihood value within this research. To measure the likelihood assessment, a ranking rating is utilized, ranging from 0 to 9 (with 9 being the highest value), based on a series of options. In this study, according to the principles outlined by the OWASP Risk Rating Methodology, there are 2 factors that need to be explored, constituting sub-processes in this research, in order to establish the likelihood value. These factors are threat agent factors and vulnerability factors.

### 3.2.4. Factors for Estimating Impact

In this phase of the process, the activity conducted is to determine the impact value within this research. Similar to measuring likelihood, impact assessment is also gauged using a ranking scale ranging from 0 to 9 (with 9 being the highest value), based on a series of options. In this study, adhering to the guidelines provided by the OWASP Risk Rating Methodology, there are 2 factors that need to be explored and constitute sub-processes in this research to establish the impact value. These factors are technical impact factors and Business Impact Factors.

### 3.2.5. Determining the Severity of the Risk

In this phase, after the complete assessment of likelihood and impact values, where the aforementioned factors have been determined through conducting a Focus Group Discussion (FGD) in collaboration with the IT team, the estimated probabilities and previously identified impacts are combined to calculate the severity level of all risks.

### 3.2.6. Result

This represents the final core process stage of this study. The overall outcomes conclusions and recommendations will be derived from the summarized document. There are two stages to be undertaken: firstly, the process of drawing overall conclusions from the research, and secondly, crafting recommendations based on the research findings.

## 4. Result and Discussion

### 4.1. Interview and Observation Result

Based on the search results, which included interviews and observations of the penetration testing outcomes, it was found that out of the 84 verification points presented in the OWASP MASVS, a total of 68 verifications were marked as "Pass." Additionally, 10 verifications were labelled as "N/A" (Not Applicable) due to limited documentation and the penetration testing team's knowledge during the testing process. Finally, 6 verifications resulted in a "Fail" status. Consequently, the next section outlines the results and findings related to the identified vulnerabilities and evidence within the digital wallet payment partner applications. The following presents the outcomes of the vulnerability identification process based on the conducted data verification searches within the application.

#### a. SAST (Static Application Security Testing)

Observations of the penetration testing using the SAST method yielded the identification of a total of 1 vulnerability within the digital wallet payment partner application. The following are some findings highlighting these existing vulnerabilities:

##### 1. The Stack Canary not Active

During the scanning process conducted with MobSF, the outcomes of the scans presented evidence that critical security features, such as stack canaries designed to thwart attempts to overwrite return addresses, are not activated. Fig. 3 displays the scanning results obtained from MobSF.

|   |                                  |                                                                                                                                                     |                                                                                                                                                                                                                                                          |
|---|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7 | lib/arm64-v8a/libtool-checker.so | True<br><b>Info</b><br>The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable. | False<br><b>High</b><br>This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries. |
| 8 | lib/arm64-v8a/libconstant.so     | True<br><b>Info</b><br>The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable. | False<br><b>High</b><br>This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries. |

Fig. 3: MobSF Scan Result for Stack Canary Value

Based on these findings, the verification results for OWASP MASVS-ID number MSTG-CODE-9 indicated a test case failure due to non-compliance with the OWASP MASVS standard.

#### b. DAST (Dynamic Application Security Testing)

The results of the penetration testing observation using the DAST method revealed the presence of 5 vulnerabilities in the digital wallet payment partner application. The following section presents the findings and evidence of these existing vulnerabilities:

##### 1. Local Storage Not Wiped After Excessive Failed Authentication.

Based on the test results and referencing the OWASP MASVS-ID MSTG-STORAGE-15, it is evident in Fig. 4 that the Local Storage within the application is not cleared of data during the authentication failure process. This situation leaves local data storage vulnerable to attacks targeting local application databases on the device.

```

daisy sprout:/data/data/com. .... mitra # ls -la *
app_textures:
total 12
drwxrwx--x 2 u0_a162 u0_a162 4096 2023-04-12 13:09 .
drwx----- 10 u0_a162 u0_a162 4096 2023-05-19 14:23 ..

app_webview:
total 4136
drwx----- 3 u0_a162 u0_a162 4096 2023-05-15 15:58 .
drwx----- 10 u0_a162 u0_a162 4096 2023-05-19 14:23 ..
-rw----- 1 u0_a162 u0_a162 4194384 2023-05-15 15:58 BrowserMetrics-spare.pma
drwx----- 5 u0_a162 u0_a162 4096 2023-05-03 11:38 Default
-rw----- 1 u0_a162 u0_a162 449 2023-05-15 15:58 pref.store
-rw----- 1 u0_a162 u0_a162 8036 2023-05-03 10:25 variations_seed
-rwx----- 1 u0_a162 u0_a162 7952 2023-05-15 15:58 variations_seed_new
-rw----- 1 u0_a162 u0_a162 0 2023-05-15 15:58 variations_stamp
-rw----- 1 u0_a162 u0_a162 25 2023-05-15 15:58 webview_data.lock

code_cache:
total 12
drwxrwx--x 2 u0_a162 u0_a162 cache 4096 2023-03-24 09:40 .
drwx----- 10 u0_a162 u0_a162 4096 2023-05-19 14:23 ..

databases:
total 124
drwxrwx--x 2 u0_a162 u0_a162 4096 2023-04-28 17:48 .
drwx----- 10 u0_a162 u0_a162 4096 2023-05-19 14:23 ..
-rw-rw---- 1 u0_a162 u0_a162 4096 2023-04-28 17:48 GenderDatabase
-rw-rw---- 1 u0_a162 u0_a162 32768 2023-04-28 17:48 GenderDatabase-shm
-rw-rw---- 1 u0_a162 u0_a162 37112 2023-04-28 17:48 GenderDatabase-wal
-rw-rw---- 1 u0_a162 u0_a162 36864 2023-05-19 14:23 .db
-rw-rw---- 1 u0_a162 u0_a162 0 2023-05-19 14:23 .db-journal

files:
total 16
drwxrwx--x 3 u0_a162 u0_a162 4096 2023-03-24 10:13 .
drwx----- 10 u0_a162 u0_a162 4096 2023-05-19 14:23 ..
drwx----- 5 u0_a162 u0_a162 4096 2023-03-24 10:13 .Fabric

no_backup:
total 16
drwxrwx--x 2 u0_a162 u0_a162 4096 2023-03-24 10:13 .
drwx----- 10 u0_a162 u0_a162 4096 2023-05-19 14:23 ..
-rw----- 1 u0_a162 u0_a162 2082 2023-03-24 10:13 com.google.InstanceId.properties
-rw----- 1 u0_a162 u0_a162 0 2023-03-24 10:13 com.google.android.gms.appid-no-backup

```

Fig. 4: Local Storage Database Still Exist after Fail Attempts Authentication

## 2. Vulnerable Third Parties Libraries and Framework

According to OWASP's criteria, a vulnerability can arise from the utilization of unidentified libraries or frameworks. In Fig. 5, it can be observed that certain libraries employed exhibit a significant level of severity, with some classified as High or Critical. The outcomes of the test cases align with the OWASP MASVS-ID number MSTG-CODE-5, highlighting that a vulnerability within a library can render an application susceptible to exploitation.

| Summary                                                                  |                                                                                                                                                                                                                                        |                                                                |                  |           |                           |
|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|------------------|-----------|---------------------------|
| Display: Showing Vulnerable Dependencies (click to show all)             |                                                                                                                                                                                                                                        |                                                                |                  |           |                           |
| Dependency                                                               | Vulnerability IDs                                                                                                                                                                                                                      | Package                                                        | Highest Severity | CVE Count | Confidence Evidence Count |
| play-services-basement-17.5.0.aar                                        |                                                                                                                                                                                                                                        | pkg.maven.com.google.android.gms:play-services-basement@17.5.0 | HIGH             | 2         | 14                        |
| inteli-core-30.0.2.jar (shaded: j9d4j9d4-1.2.17)                         | cpe 2.3 a apache:log4j:1.2.17 *****                                                                                                                                                                                                    | pkg.maven.org:log4j:log4j@1.2.17                               | CRITICAL         | 7         | Highest 13                |
| testng-7.3.0.jar                                                         | cpe 2.3 a testing:project:testing:7.3.0 *****                                                                                                                                                                                          | pkg.maven.org:testing:testing@7.3.0                            | HIGH             | 1         | Low 13                    |
| protobuf-java-3.10.0.jar                                                 | cpe 2.3 a google:protobuf:protobuf:3.10.0 *****                                                                                                                                                                                        | pkg.maven.com.google.protobuf:protobuf-java@3.10.0             | HIGH             | 4         | Highest 28                |
| okhttp-3.12.0.jar                                                        | cpe 2.3 a squaresu:okhttp:3.12.0 *****                                                                                                                                                                                                 | pkg.maven.com.squaresu.okhttp:okhttp@3.12.0                    | HIGH             | 2         | Highest 20                |
| okhttp-3.10.0.jar                                                        | cpe 2.3 a squaresu:okhttp:3.10.0 *****                                                                                                                                                                                                 | pkg.maven.com.squaresu.okhttp:okhttp@3.10.0                    | HIGH             | 2         | Highest 20                |
| inteli-core-30.0.2.jar (shaded: com.google.protobuf:protobuf-java-2.6.1) | cpe 2.3 a google:protobuf:protobuf:2.6.1 *****                                                                                                                                                                                         | pkg.maven.com.google.protobuf:protobuf-java@2.6.1              | HIGH             | 3         | Highest 14                |
| gson-2.8.6.jar                                                           | cpe 2.3 a google:gson:2.8.6 *****                                                                                                                                                                                                      | pkg.maven.com.google.code:gson:gson@2.8.6                      | HIGH             | 1         | Highest 28                |
| gson-2.8.2.jar                                                           | cpe 2.3 a google:gson:2.8.2 *****                                                                                                                                                                                                      | pkg.maven.com.google.code:gson:gson@2.8.2                      | HIGH             | 1         | Highest 28                |
| commons-compress-1.20.jar                                                | cpe 2.3 a apache:commons:commons:1.20 *****                                                                                                                                                                                            | pkg.maven.org.apache.commons:commons-compress@1.20             | HIGH             | 4         | Highest 98                |
| bcrprov-jdk15on-1.56.jar                                                 | cpe 2.3 a bouncycastle:bouncycastle:crypto:package:1.56 *****<br>cpe 2.3 a bouncycastle:bouncycastle:crypto:package:1.56 *****<br>cpe 2.3 a bouncycastle:bcprov-jdk15on-1.56 *****<br>cpe 2.3 a bouncycastle:bcprov-jdk15on-1.56 ***** | pkg.maven.org.bouncycastle:bcprov-jdk15on@1.56                 | HIGH             | 5         | Highest 38                |
| xercesImpl-2.12.0.jar                                                    | cpe 2.3 a apache:xerces:j2:2.12.0 *****                                                                                                                                                                                                | pkg.maven.org.xerces:xercesImpl@2.12.0                         | MEDIUM           | 2         | Low 72                    |
| guava-30.1-jre.jar                                                       | cpe 2.3 a google:guava:30.1 *****                                                                                                                                                                                                      | pkg.maven.com.google.guava:guava@30.1-jre                      | MEDIUM           | 1         | Highest 26                |
| testng-7.3.0.jar: suenry-3.4.1.min.js                                    |                                                                                                                                                                                                                                        | pkg.javascrt:lovenry@3.4.1.min                                 | MEDIUM           | 2         | 3                         |
| boonapi-interceptor-3.10.0.jar                                           | cpe 2.3 a squaresu:okhttp:3.10.0 *****                                                                                                                                                                                                 | pkg.maven.com.squaresu.okhttp:3toooapi                         | MEDIUM           | 1         | Highest 22                |

Fig. 5: Dependency Check Report

## 3. App Payload Communications are Visible

The payment partner application was tested by the IT Cyber Security team, in line with the verification number MASVS-ID MSTG-RESILIENCE-13 found within the application. This vulnerability arises because the JSON application payload can be exposed to attackers. The attempts made by the IT Cyber Security team provide evidence that illustrates a vulnerability or gap resulting from the lack of security measures in the application. The content of the application payload transmitted during communication with the web service will be visible, as illustrated in Fig. 6.

```

14
15 {
  "tag": "cek_update_app",
  "success": "1",
  "error": "0",
  "app_version": "44",
  "title": "UPDATED",
  "description": "UPDATED",
  "update_type": "UPDATED",
  "base_url": "https://[REDACTED]",
  "maintenance": "0",
  "title_maintenance": "",
  "body_maintenance": ""
}

```

Fig. 6: Leaked Payload

The payload that is visible in this Burp Suite can create an opening for attackers to potentially modify the JSON payload.

#### 4. The App Not Removes Sensitive Data from Views when Moved to the Background

When the application is in background mode, it doesn't implement any protection measures such as blurring or clearing content. Consequently, when the application is in the background, external parties (non-users) who access the application can view its contents. Fig. 7 provides evidence from the test results. This leads to the failure of the test case for OWASP MASVS-ID number MSTG-STORAGE-9, indicating non-compliance with the OWASP MASVS procedure.

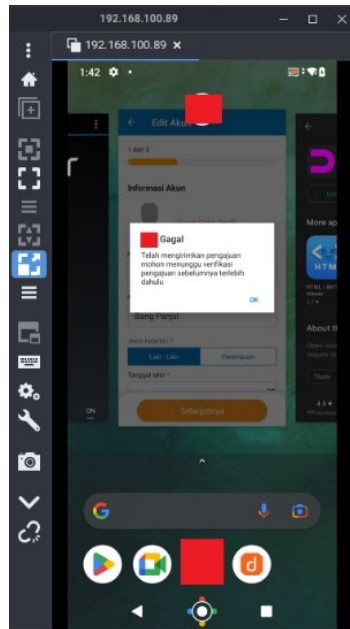


Fig. 7: Hovering Apps not Removes Sensitive Data

#### 5. App can Run on Rooted Device

One of the concerning vulnerabilities discovered in the application is its ability to run on rooted devices, even when obtained from the Google Play Store. This vulnerability is identified through the test results corresponding to OWASP MASVS-ID MSTG-RESILIENCE-1. Fig. 8 provides evidence illustrating this issue.

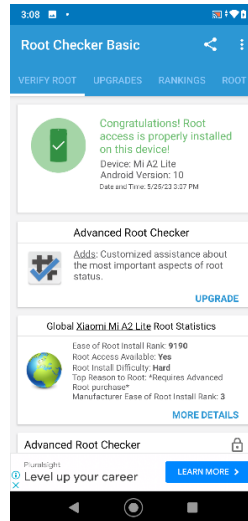


Fig. 8: Evidence Root Checker on Device

#### 4.2. Identify Risk

From the results of interviews and observations, a table of vulnerability types present in the studied mobile wallet payment partner application has been compiled. The following table outlines the vulnerability types existing in the application, along with descriptions of the threats that can exploit these vulnerabilities.

Table 1. Types of Vulnerability Risks in the Application

| No | Vulnerability Threat Type                                                      | Threat Risk Description                                                                                                                                                                                                            |
|----|--------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1  | The Stack Canary not being Active                                              | The presence of a buffer overflow attack on the application can result in program or system damage. In some cases, it can allow attackers to gain unauthorized access or take control of the system.                               |
| 2  | Local Storage Not Wiped After Excessive Failed Authentication                  | Attackers can modify users' local data, which could lead to data theft and potentially be exploited to defraud the company.                                                                                                        |
| 3  | Vulnerable Third-Party Libraries and Frameworks                                | Application malfunctions, leading to data leaks being displayed due to vulnerabilities in vulnerable libraries.                                                                                                                    |
| 4  | App Payload Communications are Visible                                         | Vulnerable to communication pathway interception between the application and web service, which can be exploited by Man-in-the-Middle attacks. Furthermore, potential attacks may involve modifying JSON payload or SQL injection. |
| 5  | The App Does Not Remove Sensitive Data from Views when Moved to the Background | Data theft can occur by accessing user information while the application is running in the background mode.                                                                                                                        |
| 6  | App Can Run on a Rooted Device                                                 | Rooted devices can compromise application security. On such devices, attacks like modifying logging and tracking network API traffic within the application can also be executed.                                                  |

#### 4.3. Factor for Estimating Likelihood

After identifying the types of vulnerabilities in the application, the next step involves assessing the likelihood value. This assessment is carried out through a Focus Group Discussion (FGD) with the company's IT team. The assessment is conducted by considering two factors: the threat agent factor and

the vulnerability factor. The process to be carried out will be explained below:

### 1. Threat Agent Factors

The first thing to do is to determine the set of factors related to the threat agents involved in order to estimate the possibility of an attack on each vulnerability that can be carried out by a group of agents. In table 2 are the criteria for the threat agent to be used.

Table 2. Criteria Threat Agent Factor Measurement

| No | Threat Agent Factors | Assessment criteria                                                                                                   |
|----|----------------------|-----------------------------------------------------------------------------------------------------------------------|
| 1  | Skill Level          | How technically skilled is this group of threat agents?                                                               |
|    |                      | No technical skills (1)                                                                                               |
|    |                      | Have some technical skills (3)                                                                                        |
|    |                      | Advance computer user (5)                                                                                             |
|    |                      | Network and programming skills (6)                                                                                    |
|    |                      | Security penetration skills (9)                                                                                       |
| 2  | Motive               | How motivated is this group of threat agents to find and exploit this vulnerability?                                  |
|    |                      | Low or no reward (1)                                                                                                  |
|    |                      | Possible reward (4)                                                                                                   |
|    |                      | High reward (9)                                                                                                       |
| 3  | Opportunity          | What resources and opportunities are required for this group of threat agents to find and exploit this vulnerability? |
|    |                      | Full access or expensive resources required (0)                                                                       |
|    |                      | Special access or resources required (4)                                                                              |
|    |                      | Some access or resources required (7)                                                                                 |
|    |                      | No access or resources required (9)                                                                                   |
| 4  | Size                 | How large is this group of threat agents?                                                                             |
|    |                      | Developers and System administrators (2)                                                                              |
|    |                      | Intranet users (4)                                                                                                    |
|    |                      | Partners (5)                                                                                                          |
|    |                      | Authenticated users (6)                                                                                               |
|    |                      | Anonymous Internet users (9)                                                                                          |

After the threat agent criteria have been determined, the next step is to assess each existing vulnerability related to the threat agent. Table 3 shows the results of the threat agent assessment of each vulnerability threat.

Table 3. Assessment of Threat Agent Factors

| No | Vulnerability Threat Type                                                      | Skill Level | Motive | Opportunity | Size |
|----|--------------------------------------------------------------------------------|-------------|--------|-------------|------|
| 1  | The Stack Canary not being Active                                              | 9           | 1      | 7           | 9    |
| 2  | Local Storage Not Wiped After Excessive Failed Authentication                  | 9           | 4      | 4           | 6    |
| 3  | Vulnerable Third-Party Libraries and Frameworks                                | 6           | 4      | 4           | 2    |
| 4  | App Payload Communications are Visible                                         | 9           | 4      | 4           | 9    |
| 5  | The App Does Not Remove Sensitive Data from Views when Moved to the Background | 1           | 1      | 9           | 6    |
| 6  | App Can Run on a Rooted Device                                                 | 3           | 1      | 4           | 6    |

After measuring the threat agent, now the next step is to calculate it the threat agent using the

following formula. The formula will be seen in Fig. 9 below. Then result will be displayed in table 4.

$$Threat\ Agent = \frac{\frac{amount\ Skill\ Level}{amount\ Vulnerability} + \frac{amount\ Motive}{amount\ Vulnerability} + \frac{amount\ Opportunity}{amount\ Vulnerability} + \frac{amount\ Size}{amount\ Vulnerability}}{4}$$

Fig. 9: Formula of Calculating Threat Agent

Table 4. Result of Total Threat Agent

| Threat Agent | Skill Level | Motive | Opportunity | Size | Total       |
|--------------|-------------|--------|-------------|------|-------------|
|              | 6.16        | 2.5    | 5.33        | 6.33 | <b>5.08</b> |

## 2. Vulnerability Factors

The first thing to do for vulnerability factors measurement is to determine the criteria for measuring vulnerability factors. It aims to estimate the possibility of vulnerabilities that are found and exploited. In table 5 are the criteria for measuring vulnerability factors.

Table 5. Criteria Vulnerability Factors Measurement

| No | Vulnerability Factors | Assessment criteria                                                                  |
|----|-----------------------|--------------------------------------------------------------------------------------|
| 1  | Ease of Discovery     | How easy is it for this group of threat agents to discover this vulnerability?       |
|    |                       | Practically impossible (1)                                                           |
|    |                       | Difficult (3)                                                                        |
|    |                       | Easy (7)                                                                             |
|    |                       | Automated tools available (9)                                                        |
| 2  | Ease of Exploit       | How motivated is this group of threat agents to find and exploit this vulnerability? |
|    |                       | Theoretical (1)                                                                      |
|    |                       | Difficult (3)                                                                        |
|    |                       | Easy (5)                                                                             |
|    |                       | Automated tools available (9)                                                        |
| 3  | Awareness             | How well known is this vulnerability to this group of threat agents?                 |
|    |                       | Unknown (1)                                                                          |
|    |                       | Hidden (4)                                                                           |
|    |                       | Obvious (6)                                                                          |
|    |                       | Public knowledge (9)                                                                 |
| 4  | Intrusion Detection   | How likely is an exploit to be detected?                                             |
|    |                       | Active detection in application (1)                                                  |
|    |                       | Logged and reviewed (3)                                                              |
|    |                       | Logged without review (8)                                                            |
|    |                       | Not logged (9)                                                                       |

After the vulnerability factors criteria have been determined, the next step is to assess each existing vulnerability related to the vulnerability factors. Table 6 shows the results of the vulnerability factors assessment of each vulnerability threat.

Table 6. Assessment of Vulnerability Factors

| No | Vulnerability Threat Type                                     | Ease of Discovery | Ease of Exploit | Awareness | Intrusion Detection |
|----|---------------------------------------------------------------|-------------------|-----------------|-----------|---------------------|
| 1  | The Stack Canary not being Active                             | 9                 | 1               | 1         | 9                   |
| 2  | Local Storage Not Wiped After Excessive Failed Authentication | 3                 | 3               | 4         | 8                   |

|   |                                                                                |   |   |   |   |
|---|--------------------------------------------------------------------------------|---|---|---|---|
| 3 | Vulnerable Third-Party Libraries and Frameworks                                | 1 | 3 | 1 | 9 |
| 4 | App Payload Communications are Visible                                         | 9 | 3 | 4 | 9 |
| 5 | The App Does Not Remove Sensitive Data from Views when Moved to the Background | 7 | 5 | 6 | 9 |
| 6 | App Can Run on a Rooted Device                                                 | 7 | 3 | 4 | 1 |

After the vulnerability factors measurement has been carried out, the next step is to measure vulnerability factors with the following formula on Fig. 10. The result will be displayed in table 7.

#### Vulnerability Factors

$$= \frac{\frac{\text{amountEase of Discovery}}{\text{amountVulnerability}} + \frac{\text{amountEase of Exploit}}{\text{amountVulnerability}} + \frac{\text{amountAwareness}}{\text{amountVulnerability}} + \frac{\text{amountIntrusion Detection}}{\text{amountVulnerability}}}{4}$$

Fig. 10: Formula of Calculation Total Vulnerability Factors

Table 7. Result of Total Vulnerability Factors

| Vulnerability Factors | Ease of Discovery | Ease of Exploit | Awareness | Intrusion Detection | Total       |
|-----------------------|-------------------|-----------------|-----------|---------------------|-------------|
|                       | 6                 | 3               | 3.33      | 7.5                 | <b>4.95</b> |

After successfully obtaining the value of the threat agent and vulnerability factor, the likelihood value can be searched. Then the following in Fig. 11 are the results of the likelihood values.

$$Likelihood = \frac{Threat Agent + Vulnerability Factor}{2} = \frac{5.08 + 4.95}{2} = \mathbf{5.01}$$

Fig. 11: Likelihood Value

#### 4.4. Factor for Estimating Impact

After identifying the likelihood values that are present, the subsequent step involves assessing the impact values. This assessment is carried out through a Focus Group Discussion (FGD) with the company's IT team. The assessment is conducted by considering two factors: Technical Impact Factors and Business Impact Factors. The process to be undertaken will be elucidated below:

##### 1. Technical Impact Factors

The first thing to do for technical impact measurement is to determine the criteria for measuring technical impact. This aims to be able to estimate the impact that will occur if the vulnerability is successfully exploited. The following in table 8 will describe the criteria that will be used to measure the technical impact.

Table 8. Criteria Technical Impact Measurement

| No | Technical Impact        | Assessment criteria                                       |
|----|-------------------------|-----------------------------------------------------------|
| 1  | Loss of Confidentiality | How much data could be disclosed and how sensitive is it? |
|    |                         | Minimal non-sensitive data disclosed (2)                  |
|    |                         | Minimal critical data disclosed (6)                       |
|    |                         | Extensive non-sensitive data disclosed (6)                |
|    |                         | Extensive critical data disclosed (7)                     |
|    |                         | All data disclosed (9)                                    |
| 2  | Loss of Integrity       | How much data could be corrupted and how damaged is it?   |

|   |                        |                                                            |
|---|------------------------|------------------------------------------------------------|
|   |                        | Minimal slightly corrupt data (1)                          |
|   |                        | minimal seriously corrupt data (3)                         |
|   |                        | Extensive slightly corrupt data (5)                        |
|   |                        | Extensive seriously corrupt data (7)                       |
|   |                        | All data totally corrupt (9)                               |
| 3 | Loss of Availability   | How much service could be lost and how vital is it?        |
|   |                        | Minimal secondary services interrupted (1)                 |
|   |                        | Minimal primary services interrupted (5)                   |
|   |                        | Extensive secondary services interrupted (5)               |
|   |                        | Extensive primary services interrupted (7)                 |
|   |                        | All services completely lost (9)                           |
| 4 | Loss of Accountability | Are the threat agents' actions traceable to an individual? |
|   |                        | Fully traceable (1)                                        |
|   |                        | Possibly traceable (7)                                     |
|   |                        | Completely anonymous (9)                                   |

After the Technical Impact criteria have been determined, the next step is to assess each existing vulnerability related to the Technical Impact. Table 9 shows the results of the technical impact assessment of each threat.

Table 9. Assessment of Technical Impact

| No | Vulnerability Threat Type                                                      | Loss of Confidentiality | Loss of Integrity | Loss of Availability | Loss of Accountability |
|----|--------------------------------------------------------------------------------|-------------------------|-------------------|----------------------|------------------------|
| 1  | The Stack Canary not being Active                                              | 2                       | 1                 | 1                    | 9                      |
| 2  | Local Storage Not Wiped After Excessive Failed Authentication                  | 6                       | 1                 | 1                    | 7                      |
| 3  | Vulnerable Third-Party Libraries and Frameworks                                | 2                       | 7                 | 5                    | 7                      |
| 4  | App Payload Communications are Visible                                         | 9                       | 1                 | 7                    | 7                      |
| 5  | The App Does Not Remove Sensitive Data from Views when Moved to the Background | 7                       | 1                 | 1                    | 9                      |
| 6  | App Can Run on a Rooted Device                                                 | 6                       | 9                 | 1                    | 7                      |

After the technical impact measurement has been carried out, the next step is to measure total of technical impact with the following formula on Fig. 12. The result will be displayed in table 10.

*Technical Impact*

$$= \frac{\text{amount Loss of Confidentiality}}{\text{amount Vulnerability}} + \frac{\text{amount Loss of Integrity}}{\text{amount Vulnerability}} + \frac{\text{amount Loss of Availability}}{\text{amount Vulnerability}} + \frac{\text{amount Loss of Accountability}}{\text{amount Vulnerability}}$$

4

Fig. 12: Formula of Calculation Total Technical Impact.

Table 10. Result of Total Technical Impact

| Technical Impact | Loss of Confidentiality | Loss of Integrity | Loss of Availability | Loss of Accountability | Total       |
|------------------|-------------------------|-------------------|----------------------|------------------------|-------------|
|                  | 5.33                    | 3.33              | 2.66                 | 7.66                   | <b>4.74</b> |

## 2. Business Impact Factors

The first thing that must be done for measuring business impact is to determine the criteria for measuring business impact. This aims to estimate the business impact that will be received from exploiting existing vulnerabilities. Therefore table 11 shows the measurement criteria to be searched for

Table 11. Criteria Business Impact Measurement

| No | Business Impact   | Assessment criteria                                                        |
|----|-------------------|----------------------------------------------------------------------------|
| 1  | Financial damage  | How much financial damage will result from an exploit?                     |
|    |                   | Less than the cost to fix vulnerability (1)                                |
|    |                   | Minor effect on annual profit (3)                                          |
|    |                   | Significant effect on annual profit (7)                                    |
|    |                   | Bankruptcy (9)                                                             |
| 2  | Reputation damage | Would an exploit result in reputation damage that would harm the business? |
|    |                   | Minimal damage (1)                                                         |
|    |                   | Loss of major accounts (4)                                                 |
|    |                   | Loss of goodwill (5)                                                       |
|    |                   | Brand damage (9)                                                           |
| 3  | Non-compliance    | How much exposure does non-compliance introduce?                           |
|    |                   | Minor violation (2)                                                        |
|    |                   | Clear violation (5)                                                        |
|    |                   | High profile violation (7)                                                 |
| 4  | Privacy violation | How much personally identifiable information could be disclosed?           |
|    |                   | One individual (3)                                                         |
|    |                   | Thousands of people (7)                                                    |
|    |                   | Millions of people (9)                                                     |

After the business impact criteria have been determined, the next step is to assess each existing vulnerability related to the business impact. Table 12 shows the results of the business impact assessment of each threat.

Table 12. Assessment of Business Impact

| No | Vulnerability Threat Type                                                      | Financial damage | Reputation damage | Non-compliance | Privacy violation |
|----|--------------------------------------------------------------------------------|------------------|-------------------|----------------|-------------------|
| 1  | The Stack Canary not being Active                                              | 1                | 5                 | 2              | 3                 |
| 2  | Local Storage Not Wiped After Excessive Failed Authentication                  | 1                | 1                 | 2              | 3                 |
| 3  | Vulnerable Third-Party Libraries and Frameworks                                | 3                | 5                 | 7              | 7                 |
| 4  | App Payload Communications are Visible                                         | 3                | 1                 | 5              | 7                 |
| 5  | The App Does Not Remove Sensitive Data from Views when Moved to the Background | 1                | 4                 | 5              | 3                 |
| 6  | App Can Run on a Rooted Device                                                 | 7                | 5                 | 7              | 3                 |

After measuring the business impact, the next step is to measure the total business impact with the

following formula in Fig. 13. The results are shown in table 13.

*Business Impact*

$$= \frac{\text{amountFinancial Damage}}{\text{amountVulnerability}} + \frac{\text{amountReputation Damage}}{\text{amountVulnerability}} + \frac{\text{amountNon-compliance}}{\text{amountVulnerability}} + \frac{\text{amountPrivacy violation}}{\text{amountVulnerability}}$$

4

Fig. 13: Formula of Calculation Total Business Impact.

Table 13. Result of Total Business

| Business Impact | Financial Damage | Reputation Damage | Non-compliance | Privacy Violation | Total |
|-----------------|------------------|-------------------|----------------|-------------------|-------|
|                 | 2.66             | 3.5               | 4.66           | 4.33              | 3.78  |

After successfully obtaining the value of the technical impact and business impact, the impact value can be searched. Then the following in Fig.14 are the results of the impact values.

$$impact = \frac{Technical Impact + Business Impact}{2} = \frac{4.74 + 3.78}{2} = 4.26$$

Fig. 14: Impact Value

#### 4.5. Determining Severity of the Risk

Once the likelihood and impact have been determined, the subsequent step involves assigning a risk severity for each value. Table 14 illustrates the level rating scale that will be applied to both likelihood and impact

Table 14. Likelihood and Impact Level Scale

| Likelihood and Impact Level Scale |        |
|-----------------------------------|--------|
| 0 < 3                             | LOW    |
| 3 < 6                             | MEDIUM |
| 6 < 9                             | HIGH   |

Based on the values obtained in the previous assessment, the likelihood value obtained was 5.01. This indicates that the likelihood falls within the MEDIUM level on the scale. The obtained impact value is 4.26, corresponding to a MEDIUM impact level. Subsequently, an overall severity matrix is constructed and will be presented in table 15 below.

Table 15. Overall Severity Matrix

| IMPACT     | HIGH   | MEDIUM | HIGH   | CRITICAL |
|------------|--------|--------|--------|----------|
|            | MEDIUM | LOW    | MEDIUM | HIGH     |
|            | LOW    | NOTE   | LOW    | MEDIUM   |
|            |        | LOW    | MEDIUM | HIGH     |
| LIKELIHOOD |        |        |        |          |

From the above results, it can be observed that the intersection between the MEDIUM likelihood and MEDIUM impact levels yields a MEDIUM level. Consequently, the overall severity risk value obtained is MEDIUM.

#### 4.6. Deciding What to Fix

So, the last step is to provide a solution to this digital wallet payment partner application, related to fixing the vulnerabilities that exist in the application. In table 16 are the solutions that need to be applied to the vulnerabilities found.

Table 16. Solution to Fix in Application

| No | Vulnerability Threat                                                           | Risk Value | Solution                                                                                                                                                                                                                                                                                                                             |
|----|--------------------------------------------------------------------------------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1  | The Stack Canary not being Active                                              | LOW        | Implement the improvement by researching how to enable the stack protector. For this solution, it can be carried out during the application build process by activating stack protection. Further information on this can be explored by the mobile application development team.                                                    |
| 2  | Local Storage Not Wiped After Excessive Failed Authentication                  | LOW        | Carry out the deletion of local data storage, such as shared preferences, after multiple unsuccessful authentication attempts. Set a maximum time limit for user authentication attempts and perform data storage deletion accordingly.                                                                                              |
| 3  | Vulnerable Third-Party Libraries and Frameworks                                | MEDIUM     | Perform verification and authenticity checks on libraries by examining their original library repository (such as GitHub) to understand the library's development progress. Additionally, updating libraries to their latest versions can reduce vulnerabilities and replace libraries that lack new updates from their maintainers. |
| 4  | App Payload Communications are Visible                                         | HIGH       | Incorporate security measures by encrypting the JSON data during application communication with web services. Furthermore, endeavour to enhance security by implementing SSL pinning for additional protection.                                                                                                                      |
| 5  | The App Does Not Remove Sensitive Data from Views when Moved to the Background | MEDIUM     | In this improvement, apply additional code or algorithms that can create a blur effect or white screen when the application runs in the background. Use this code on pages that contain sensitive user data, such as the registration or user profile editing pages.                                                                 |
| 6  | App Can Run on a Rooted Device                                                 | MEDIUM     | Improvements can be made by adding algorithmic code or libraries that can detect and validate the prevention of rooted devices within the application. This can be achieved by triggering alert dialogs and preventing the application from running, or implementing a forced exit in the application running on rooted devices.     |

From the table above, it is known that from the vulnerability assessment results, the highest risk value is associated with vulnerability number 4. Vulnerability number 4 describes the risk value related to the presence of payload that can be viewed by attackers in the application, which is categorized as HIGH. Therefore, this vulnerability point becomes the priority for the mitigation process that needs to be carried out first in the digital wallet payment partner application. Meanwhile, the mitigation process for other vulnerabilities with MODERATE and LOW risk values will follow subsequently.

## 5. Conclusion

### 5.1. Research Conclusion

Using the OWASP method, researchers succeeded in determining the security values of mobile applications based on the OWASP MASVS guidelines. With the OWASP Risk Rating Methodology,

the search for risk values can be conducted, providing the outcomes of all the verification processes that have been conducted. The researchers identified 6 vulnerabilities in the Android mobile-based digital wallet payment partner application owned by one of the digital wallet companies in Indonesia. Based on the comprehensive evaluation results, the overall risk rating that was established resulted in a MEDIUM value. Subsequently, mitigation suggestions were provided for the digital wallet payment partner application, which needs to implement improvements according to the recommended solutions to prevent potential attacks exploiting the identified vulnerabilities in the application. Thus, from this research, it can be concluded that the use of the OWASP method can address all research questions summarized as follows:

- RQ1, which is the process of vulnerability identification, can be sought by using the verification process provided by OWASP MASVS, which includes testing guidelines provided by OWASP MASTG.
- RQ2 has been determined through the search and identification of risks that have been conducted.
- RQ3, regarding risk assessment, can be determined using the OWASP Risk Rating Methodology. In this study, the severity risk value is MEDIUM, and the mitigation solution process, the value of each vulnerability, and the priority of mitigation have also been explained in the sixth step of the results and discussion.

## 5.2. Suggestion

The suggestion that can be given from this research is the need for testing by accessing or viewing the contents of confidential documentation files or implementing more secure access rights on the application system that uses third-party services in the company, so that all test cases can be conducted and do not have an N/A (Not Applicable) rating in OWASP MASVS. Additionally, additional knowledge is required to detail the observation of penetration and provide better solutions to prevent vulnerabilities existing in the application.

## 5.3. Limitation

The limitations of this research include the potential addition of features and advancements in application technology in the future, rendering the current security assessment obsolete. Additionally, the data search conducted using a qualitative method approach results in subjectivity. Thus, there is a possibility that risks might not be defined in accordance with actual conditions due to the subjective perspective of the author. Furthermore, as OWASP continues to enhance and update verification versions and the associated guidelines, such as OWASP MASVS or MSTG, the verification process might become less relevant in the future.

## 5.4. Future Work

Future work can be recommended, particularly for enterprises, by conducting a reassessment of application security after applying fixes to the application. This aims to ascertain the effectiveness and demonstrate that security assessments using the OWASP method can indeed be conducted and implemented within companies.

## References

- Ali, B., Iris, R., Rajiv, S., & Joseph, A. (2019). "Mobile technology identity and self-efficacy: Implications for the adoption of clinically supported Mobile Health Apps," *International Journal of Information Management*, vol. 49, pp. 58–68.
- Mei-Ling, Y., Ming-Chuen, C., & Chun-Cheng, H. (2018) "The Kano Model Analysis of features for mobile security applications," *Computers & Security*, vol. 78, pp. 336–346.
- Björn, L & Niklas, M, "Defining information security," *Science and Engineering Ethics*, vol. 25, no. 2, pp. 419–441, 2017. doi:10.1007/s11948-017-9992-1

Ruth, C., "Over 60% of Android Apps Have Security Flaws - Atlas VPN," atlasVPN, <https://atlasvpn.com/blog/over-60-of-android-apps-have-security-vulnerabilities> (accessed Mar. 26, 2023).

Mediaindonesia.com, "Serangan Siber Ke Sektor Keuangan Dan Perbankan di Indonesia Lebih banyak dari RERATA global," Berita Terkini Hari ini Indonesia dan Dunia - Media Indonesia, <https://mediaindonesia.com/teknologi/517197/serangan-siber-ke-sektor-keuangan-dan-perbankan-di-indonesia-lebih-banyak-dari-rerata-global> (accessed Mar. 26, 2023).

Guardsquare, "Mobile App Security Statistics You Must know," Guardsquare, <https://www.guardsquare.com/blog/4-surprising-stats-prove-need-mobile-app-security> (accessed Mar. 26, 2023).

Paul, F., Michael, W., & Deborah, R. (2021). "A Principlist Framework for cybersecurity ethics," *Computers & Security*, vol. 109, p. 102382.

Positive Technologies, "Vulnerabilities and threats in mobile applications, 2019," ptsecurity.com, <https://www.ptsecurity.com/ww-en/analytics/mobile-application-security-threats-and-vulnerabilities-2019/> (accessed Mar. 26, 2023).

Pradeo, Global Mobile Security Report 2022, <https://www.pradeo.com/en-US/datasheet/mobile-security-threat-report> (accessed Mar. 26, 2023).

Anthony, V., & Yohan, M. (2021). "Peeking and testing broken object level authorization vulnerability onto e-commerce and E-banking mobile applications," *Procedia Computer Science*, vol. 179, pp. 962–965, 2021.

OWASP, "Owasp Mobile top 10," OWASP Mobile Top 10 | OWASP Foundation, <https://owasp.org/www-project-mobile-top-10/> (accessed Mar. 26, 2023).

OWASP, "Owasp Risk Rating Methodology," OWASP Risk Rating Methodology | OWASP Foundation, [https://owasp.org/www-community/OWASP\\_Risk\\_Rating\\_Methodology](https://owasp.org/www-community/OWASP_Risk_Rating_Methodology) (Accessed: 26-Mar-2023).

Francisco, J. R., Ángel, J.V., Jorge, R., Joaquín, L., & Alejandro, C. (2019). "A framework to secure the development and auditing of SSL pinning in mobile applications: The case of Android devices," *Entropy*, vol. 21, no. 12, p. 1136, 2019.

Carlos, H., Bernhard, M., Sven, S., & Jeroen, W., "Owasp MASVS," *OWASP MASVS - OWASP Mobile Application Security*, <https://mas.owasp.org/MASVS/> (Accessed: 26-Mar-2023).

Carlos, H., Bernhard, M., Sven, S., & Jeroen, W., "Owasp MASTG," *OWASP MASTG - OWASP Mobile Application Security*, <https://mas.owasp.org/MASTG/> (Accessed: 26-Mar-2023).

Kai, Q., Reza, M. P., & Dan, L. (2018). "Owasp Risk Analysis Driven Security Requirements Specification for secure Android Mobile Software Development," *2018 IEEE Conference on Dependable and Secure Computing (DSC)*.

"Mobile Application Security Testing," Mobile Application Security Testing - OWASP MASTG, <https://mobile-security.gitbook.io/mobile-security-testing-guide/overview/0x04b-mobile-app-security-testing> (accessed May 21, 2023).

Handayani, T., & Novitasari, A. (2020). Digital Wallet as a transaction media in the community. IOP Conference Series: Materials Science and Engineering, 879(1), 012001. <https://doi.org/10.1088/1757-899x/879/1/012001>

John, W., C., & J, D. C. (2018). Research design: Qualitative, quantitative, and mixed methods approaches. SAGE Publications, Inc.

Robert, K., Y. (2018). Case study research and applications: Design and methods. SAGE.

Seçil, T., & Kwestan, K. A. (2021). An overview of qualitative research and focus group discussion. *International Journal of Academic Research in Education*, 7(1), 1–15. <https://doi.org/10.17985/ijare.866762>

Noppanat, P., & Vasaka, V. (2018). Android forensic and Security Assessment for hospital and stock-and-trade applications in Thailand. 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE). <https://doi.org/10.1109/jcsse.2018.8457347>

Carla, S., W. (2023, March 31). DOMPET DIGITAL NAIK DAUN, MEMBETOT MINAT KALA PANDEMI. Dompert Digital Naik Daun, Membetot Minat Kala Pandemi. <https://www.bi.go.id/id/bi-institute/BI-Epsilon/Pages/Dompert-Digital--Naik-Daun,-Membetot-Minat-Kala-Pandemi.aspx>

Kevin, K., & Dennis, G. (2019). Guillou-quisquater protocol for user authentication based on Zero knowledge proof. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 17(2), 826. <https://doi.org/10.12928/telkomnika.v17i2.11754>

Ha, L. V., On, P. V., & Hoa, N. N. (2020). Information Security Risk Management by a Holistic Approach: a Case Study for Vietnamese e-Government. *International Journal of Computer Science and Network Security (IJCSNS)*, 20, 72–82.

Robertus, A., & Ditdit, N., U. (2023). Business Continuity Management Framework In Electronic System Provider (ESP) Startup Company. *Journal of System and Management Sciences*, 13, 322-343. doi:10.33168/JSMS.2023.0118