Threat Modeling Report

Project: halodoc-android-master

Property	Value
Project Name	halodoc-android-master
Analysis Date	2025-10-18T21:45:20.860150
Methodology	STRIDE
Total Findings	4
Risk Level	HIGH

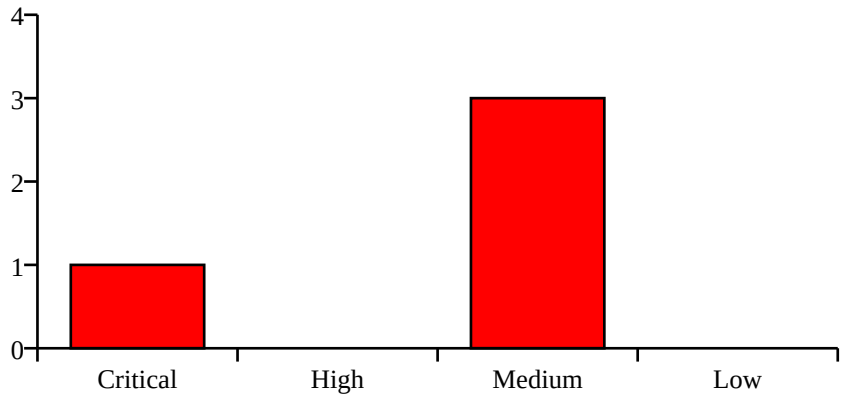
Executive Summary

This security assessment of **halodoc-android-master** identified **4** potential security threats using the STRIDE methodology. The overall risk level is assessed as **HIGH**.

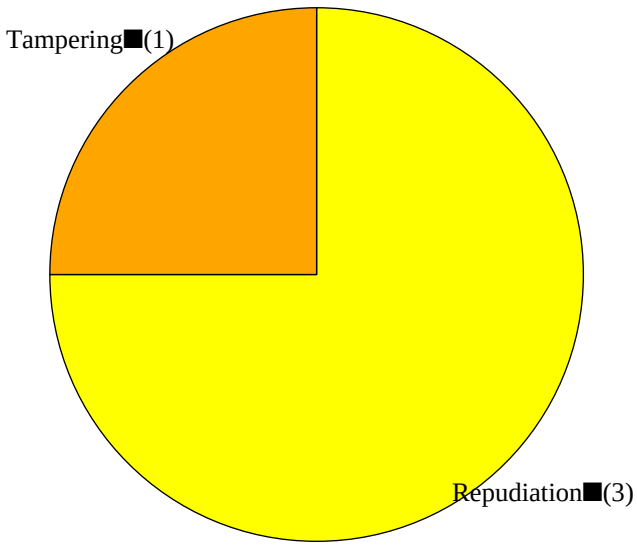
Key Findings:
• Critical vulnerabilities: 1
• High-severity issues: 0
• Medium-priority concerns: 3
• Low-priority items: 0

Immediate attention is required for all critical and high-severity vulnerabilities to prevent potential security breaches.

Threat Severity Distribution



STRIDE Category Distribution



STRIDE Methodology Overview

STRIDE is a threat modeling methodology developed by Microsoft that categorizes security threats into six main areas:

S - Spoofing Identity: Impersonating someone or something else to gain unauthorized access
T - Tampering with Data: Malicious modification of data or code
R - Repudiation: Users denying they performed an action without the system being able to prove otherwise
I - Information Disclosure: Exposure of information to individuals who shouldn't have access
D - Denial of Service: Attacks that deny or degrade service for legitimate users
E - Elevation of Privilege: A user gains capabilities without proper authorization

Each identified threat is categorized into one of these areas and assessed for severity and impact.

Project Architecture Analysis

Code Analysis Summary:
• Files analyzed: 3
• Programming languages: Kotlin
• Threat detection patterns: STRIDE-based security analysis
• Analysis depth: Source code static analysis with context awareness

Detailed Security Findings

Finding #1: Command injection

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-78
Confidence Score	0.90
File Location	halodoc-android-master/app/src/main/java/com/linkdokter/halodoc/android/util/AppHelp
Attack Vector	Data modification, code injection, integrity violations

Description:

Command injection detected in Kotlin code

Code Evidence:

```
DisplayMetrics().also { display.getRealMetrics(it) } } else { >>>
Resources.getSystem().displayMetrics } return metrics.heightPixels
```

Proof of Concept:

Steps to Reproduce:
1. Identify data modification point
2. Analyze input validation mechanisms
3. Craft malicious payload
4. Submit payload to modify data/code
5. Verify successful tampering

Impact: Data corruption, system integrity compromise

Remediation:

Use parameterized commands, input validation, and avoid shell execution

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #2: Logging disabled

Property	Details
Severity	Medium
STRIDE Category	Repudiation
CWE ID	CWE-778
Confidence Score	0.70
File Location	halodoc-android-master/app/src/main/java/com/linkdokter/halodoc/android/AppUserLo
Attack Vector	Log tampering, evidence destruction, transaction denial

Description:

Logging disabled detected in Kotlin code

Code Evidence:

```
private val transporterLogger: HalodocTransporterLogger =
HalodocTransporterLogger.getInstance()) { >>> private var loginState :
LoginState? = null private val applicationScope =
CoroutineScope(SupervisorJob() + Dispatchers.Main)
```

Proof of Concept:

Steps to Reproduce:
1. Review the identified vulnerability in source code
2. Analyze potential attack vectors
3. Develop exploitation methodology
4. Test vulnerability in controlled environment
5. Document impact and exploitability

Impact: Security compromise as per STRIDE category

Remediation:

Implement comprehensive logging and audit trails

Business Impact:

Limited audit trail issues, minor compliance gaps

Finding #3: Logging disabled

Property	Details
Severity	Medium
STRIDE Category	Repudiation
CWE ID	CWE-778
Confidence Score	0.80
File Location	halodoc-android-master/app/src/main/java/com/linkdokter/halodoc/android/AppUserLo
Attack Vector	Log tampering, evidence destruction, transaction denial

Description:

Logging disabled detected in Kotlin code

Code Evidence:

```
companion object { >>> private var INSTANCE: AppUserLoginStateObserver? = null
@JvmStatic fun getInstance(floresModule: FloresModule,
```

Proof of Concept:

Steps to Reproduce:
1. Review the identified vulnerability in source code
2. Analyze potential attack vectors
3. Develop exploitation methodology
4. Test vulnerability in controlled environment
5. Document impact and exploitability

Impact: Security compromise as per STRIDE category

Remediation:

Implement comprehensive logging and audit trails

Business Impact:

Limited audit trail issues, minor compliance gaps

Finding #4: Logging disabled

Property	Details
Severity	Medium
STRIDE Category	Repudiation

CWE ID	CWE-778
Confidence Score	0.80
File Location	halodoc-android-master/app/src/main/java/com/linkdokter/halodoc/android/logger/Halo
Attack Vector	Log tampering, evidence destruction, transaction denial

Description:

Logging disabled detected in Kotlin code

Code Evidence:

```
when { >>> !logBody || requestBody == null -> { logger.logInTimber("--> END
${request.method}") }
```

Proof of Concept:

Steps to Reproduce:
1. Review the identified vulnerability in source code
2. Analyze potential attack vectors
3. Develop exploitation methodology
4. Test vulnerability in controlled environment
5. Document impact and exploitability

Impact: Security compromise as per STRIDE category

Remediation:

Implement comprehensive logging and audit trails

Business Impact:

Limited audit trail issues, minor compliance gaps

Remediation Summary

Remediation Priority Matrix

■ IMMEDIATE (0-7 days) - Critical Issues: 1
Critical vulnerabilities pose immediate risk to business operations and must be addressed urgently. Recommended actions: Emergency patches, temporary mitigations, incident response preparation.

■ HIGH PRIORITY (1-4 weeks) - High Severity: 0
High-severity issues should be addressed in the next sprint cycle. Recommended actions: Security patches, code reviews, testing validation.

■ MEDIUM PRIORITY (1-3 months) - Medium Severity: 3
Medium-severity issues can be addressed in regular development cycles. Recommended actions: Security improvements, best practice implementation, monitoring enhancement.

Implementation Guidelines:
• Establish security champion within development team
• Implement security testing in CI/CD pipeline
• Conduct regular security code reviews
• Provide security training for developers
• Monitor for new vulnerabilities and threat intelligence
• Regular penetration testing and security assessments