Comprehensive Threat Modeling Report

All Projects Security Analysis

Metric	Value
Total Projects Analyzed	4
Total Threats Found	106
Critical Threats	58
High Threats	1
Medium Threats	47
Low Threats	0
Analysis Date	2025-10-18 21:50:16
Methodology	STRIDE

Overall Risk Assessment

Overall Risk Level: CRITICAL

Key Findings:
• 58 Critical vulnerabilities requiring immediate attention
• 1 High-severity issues needing prompt remediation
• 47 Medium-priority security concerns
• Analysis covers 4 projects using STRIDE methodology

Primary Threat Categories:
• Tampering: 58 threats
• Information_Disclosure: 44 threats
• Repudiation: 3 threats
• Spoofing: 1 threats

STRIDE Methodology Overview

STRIDE is a threat modeling methodology that categorizes security threats into six main categories:

S - Spoofing: Impersonating someone or something else
T - Tampering: Modifying data or code
R - Repudiation: Claiming to have not performed an action
I - Information Disclosure: Exposing information to unauthorized individuals
D - Denial of Service: Denying or degrading service to users
E - Elevation of Privilege: Gaining capabilities without proper authorization

This analysis identified threats across all STRIDE categories with detailed evidence and mitigation strategies.

STRIDE Category	Total Threats	Percentage	Risk Level
Repudiation	3	2.8%	Low
Tampering	58	54.7%	High
Information_Disclosure	44	41.5%	High
Spoofing	1	0.9%	Low

Detailed Project Analysis

Project: halodoc-android-master

Total Findings: 4
Analysis Date: 2025-10-18T21:45:20.860150
Risk Level: HIGH

Severity	Count
Critical	1
Medium	3

Critical Findings:

1. Command injection
Severity: Critical
STRIDE Category: Tampering
CWE: CWE-78
File:
halodoc-android-master/app/src/main/java/com/linkdokter/halodoc/android/util/AppHelper.kt:93
Description: Command injection detected in Kotlin code
Mitigation: Use parameterized commands, input validation, and avoid shell execution
Code Evidence:

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

2. Logging disabled
Severity: Medium
STRIDE Category: Repudiation
CWE: CWE-778
File: halodoc-android-master/app/src/main/java/com/linkdokter/halodoc/android/AppUserLoginStateObserver.kt:24
Description: Logging disabled detected in Kotlin code
Mitigation: Implement comprehensive logging and audit trails
Code Evidence:

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

3. Logging disabled
Severity: Medium
STRIDE Category: Repudiation
CWE: CWE-778
File: halodoc-android-master/app/src/main/java/com/linkdokter/halodoc/android/AppUserLoginStateObserver.kt:82
Description: Logging disabled detected in Kotlin code
Mitigation: Implement comprehensive logging and audit trails
Code Evidence:

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

4. Logging disabled
Severity: Medium

STRIDE Category: Repudiation
CWE: CWE-778
File: halodoc-android-master/app/src/main/java/com/linkdokter/halodoc/android/logger/HalodocNetworkInterceptor.kt:151
Description: Logging disabled detected in Kotlin code
Mitigation: Implement comprehensive logging and audit trails
Code Evidence:

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

Project: hospitalportal-master

Total Findings: 82
Analysis Date: 2025-10-18T21:45:22.501142
Risk Level: CRITICAL

Severity	Count
Critical	55
Medium	27

Critical Findings:

1. Code injection via eval
Severity: Critical
STRIDE Category: Tampering
CWE: CWE-95
File: hospitalportal-master/HospitalPortal/Content/js/jquery.validate.unobtrusive.min.js:5
Description: Code injection via eval detected in JavaScript code
Mitigation: Avoid dynamic code execution, use safe alternatives

2. Code injection via eval
Severity: Critical
STRIDE Category: Tampering
CWE: CWE-95
File: hospitalportal-master/HospitalPortal/Content/js/jquery.validate.unobtrusive.min.js:5
Description: Code injection via eval detected in JavaScript code
Mitigation: Avoid dynamic code execution, use safe alternatives

3. Code injection via exec
Severity: Critical
STRIDE Category: Tampering
CWE: CWE-95
File: hospitalportal-master/HospitalPortal/Content/js/jquery.validate.unobtrusive.min.js:5
Description: Code injection via exec detected in JavaScript code
Mitigation: Avoid dynamic code execution, use safe alternatives

4. Code injection via eval
Severity: Critical
STRIDE Category: Tampering
CWE: CWE-95
File: hospitalportal-master/HospitalPortal/Content/js/EditTable.js:494
Description: Code injection via eval detected in JavaScript code
Mitigation: Avoid dynamic code execution, use safe alternatives
Code Evidence:

```
/* If it is string assume it is json. */ if (String == data.constructor) { >>>
eval('var json = ' + data); } else { /* Otherwise assume it is a hash already. */
```

5. Code injection via exec
Severity: Critical
STRIDE Category: Tampering
CWE: CWE-95
File: hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.flot.js:32
Description: Code injection via exec detected in JavaScript code
Mitigation: Avoid dynamic code execution, use safe alternatives

Project: halodoc-ios-master

Total Findings: 4
Analysis Date: 2025-10-18T21:45:23.124275
Risk Level: HIGH

Severity	Count
Critical	2
Medium	2

Critical Findings:

1. Code injection via eval
Severity: Critical
STRIDE Category: Tampering
CWE: CWE-95
File: halodoc-ios-master/crash_monkey_result/result_000/result_view.js:42
Description: Code injection via eval detected in JavaScript code
Mitigation: Avoid dynamic code execution, use safe alternatives
Code Evidence:

```
index: i }); >>> return eval(log.message); }); return img.src = log.screen_image
+ '.png';
```

2. Code injection via eval
Severity: Critical
STRIDE Category: Tampering
CWE: CWE-95
File: halodoc-ios-master/crash_monkey_result/result_001/result_view.js:42
Description: Code injection via eval detected in JavaScript code
Mitigation: Avoid dynamic code execution, use safe alternatives
Code Evidence:

```
index: i }); >>> return eval(log.message); }); return img.src = log.screen_image
+ '.png';
```

3. Information disclosure in errors
Severity: Medium
STRIDE Category: Information_Disclosure
CWE: CWE-209
File: halodoc-ios-master/crash_monkey_result/UIAutoMonkey.js:165
Description: Information disclosure in errors detected in JavaScript code
Mitigation: Implement proper error handling without information disclosure
Code Evidence:

```
if (!event) { UIALogger.logMessage("Attempted to " + name) >>> throw new
Error("Attempted to fire an undefined event '" + name + "'!") }
event.apply(this);
```

4. Insecure HTTP usage
Severity: Medium
STRIDE Category: Information_Disclosure
CWE: CWE-319
File: halodoc-ios-master/crash_monkey_result/UIAutoMonkey.js:2
Description: Insecure HTTP usage detected in JavaScript code
Mitigation: Use HTTPS/TLS for all communications
Code Evidence:

```
>>> // Copyright (c) 2013 Jonathan Penn (http://cocoamanifest.net/) // Permission is hereby granted, free of charge, to any person obtaining a copy
```

Project: batavia-client-master

Total Findings: 16
Analysis Date: 2025-10-18T21:45:23.210269
Risk Level: MEDIUM

Severity	Count
High	1
Medium	15

Critical Findings:

1. Hardcoded password
Severity: High
STRIDE Category: Spoofing
CWE: CWE-798
File: batavia-client-master/src/app/modules/login/login.component.ts:19
Description: Hardcoded password detected in TypeScript code
Mitigation: Use environment variables or secure credential management
Code Evidence:

```
private ngUnsubscribe$ = new Subject(); username = ''; >>> password = '';
showErrorMsg = false; returnUrl: string;
```

2. Insecure HTTP usage
Severity: Medium
STRIDE Category: Information_Disclosure
CWE: CWE-319
File: batavia-client-master/src/polyfills.ts:33
Description: Insecure HTTP usage detected in TypeScript code
Mitigation: Use HTTPS/TLS for all communications
Code Evidence:

```
/** * Required to support Web Animations `@angular/animation`. >>> * Needed for:
All but Chrome, Firefox and Opera. http://caniuse.com/#feat=web-animation **/
```

3. Insecure HTTP usage
Severity: Medium
STRIDE Category: Information_Disclosure
CWE: CWE-319
File: batavia-client-master/src/environments/environment.ts:12
Description: Insecure HTTP usage detected in TypeScript code
Mitigation: Use HTTPS/TLS for all communications
Code Evidence:

```
VERSION: require('../../package.json').version, // baseApiUrl:
'https://controlcenter.stage.halodoc.com', >>> baseApiUrl:
'http://localhost:4200', baseAuthUrl: '/api', googleMapApi: {
```

4. Insecure HTTP usage
Severity: Medium
STRIDE Category: Information_Disclosure
CWE: CWE-319
File: batavia-client-master/src/environments/environment.ts:29
Description: Insecure HTTP usage detected in TypeScript code
Mitigation: Use HTTPS/TLS for all communications
Code Evidence:

```
mfeConfig: { bataviaCmsMfe: { >>> remoteEntry:
'http://localhost:4200/cms-mfe/remoteEntry.js', exposedModule:
'CmsMfeWrapperModule', },
```

5. Insecure HTTP usage
Severity: Medium
STRIDE Category: Information_Disclosure
CWE: CWE-319
File: batavia-client-master/src/environments/environment.ts:33
Description: Insecure HTTP usage detected in TypeScript code
Mitigation: Use HTTPS/TLS for all communications
Code Evidence:

```
}, bataviaLabsMfe: { >>> remoteEntry:
'http://localhost:4200/labs-mfe/remoteEntry.js', exposedModule:
'LabsMfeWrapperModule', },
```

Security Recommendations

Priority Recommendations:

■ **CRITICAL:** 58 critical vulnerabilities require immediate attention.
• Implement comprehensive input validation and parameterized queries
• Review all dynamic code execution patterns
■■ **HIGH:** 1 high-severity issues need prompt remediation.
• Review information disclosure vulnerabilities
• Implement proper error handling
• Upgrade to secure communication protocols
• Strengthen authentication mechanisms
• Remove hardcoded credentials

General Security Improvements:
• Implement automated security testing in CI/CD pipeline
• Conduct regular security code reviews
• Provide security training for development teams
• Establish incident response procedures

Implementation Roadmap

Phase 1 - Immediate Actions (0-30 days):
• Address all Critical vulnerabilities
• Remove hardcoded credentials and API keys
• Implement input validation for code injection vulnerabilities
• Enable HTTPS for all communications

Phase 2 - Short-term Improvements (1-3 months):
• Remediate High severity vulnerabilities
• Implement comprehensive logging and monitoring
• Upgrade weak cryptographic algorithms
• Conduct security code reviews

Phase 3 - Long-term Security Program (3-12 months):
• Address Medium severity vulnerabilities
• Implement automated security testing
• Security awareness training for developers
• Regular threat modeling and security assessments

Appendix A: STRIDE Threat Categories

Spoofing Identity
Threats involving impersonation of users, processes, or systems.
Common examples: Credential theft, session hijacking, identity fraud

Tampering with Data
Unauthorized modification of data or code.
Common examples: SQL injection, XSS, data corruption, code injection

Repudiation
Users denying they performed an action without the system being able to prove otherwise.
Common examples: Insufficient logging, missing audit trails, log tampering

Information Disclosure
Exposure of information to individuals who are not supposed to have access to it.
Common examples: Data leaks, privacy violations, insecure communications

Denial of Service
Attacks that deny or degrade service for users.
Common examples: Resource exhaustion, DDoS attacks, system overload

Elevation of Privilege
A user gains capabilities without proper authorization.
Common examples: Buffer overflows, privilege escalation, configuration errors