Mobile Security

Mobile Security

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Mobile Threat Model

Threat Actor	Goal	Attack Vector	Likelihood	Impact	Mitigation
Malicious app on device	Steal auth tokens	Shared storage access	Medium	High	Secure storage (Keychain/Keystore)
Network interceptor (MITM)	Intercept API calls	Rogue WiFi, proxy	Medium	High	Certificate pinning
Reverse engineer	Extract API keys, business logic	APK/IPA decompilation	Medium	Medium	Code obfuscation, no hardcoded secrets
Stolen/lost device	Access user data	Physical access	High	High	Biometric lock, data encryption, remote wipe
Rooted/jailbroken device	Bypass controls	OS privilege escalation	Low	High	Root/jailbreak detection

Threat Actor	Goal	Attack Vector	Likelihood	Impact	Mitigation
Malicious insider	Access user data	Source code access	Low	High	Secrets in vault, no PII in logs

Assets to protect:

1. Auth tokens (access + refresh)
2. User PII (name, email, payment info)
3. API keys and secrets
4. Cached sensitive data

---

2. Authentication

2.1 Biometric Authentication

Implementation: `{{expo-local-authentication | react-native-biometrics}}`

```
import * as LocalAuthentication from 'expo-local-authentication';

export async function authenticateWithBiometrics(): Promise<boolean> {
  const hasHardware = await LocalAuthentication.hasHardwareAsync();
  const isEnrolled = await LocalAuthentication.isEnrolledAsync();

  if (!hasHardware || !isEnrolled) {
    return promptPINFallback();
  }

  const result = await LocalAuthentication.authenticateAsync({
    promptMessage: 'Verify your identity',
    fallbackLabel: 'Use PIN',
    cancelLabel: 'Cancel',
    disableDeviceFallback: false,
  });

  return result.success;
}
```

Use cases for biometric auth:

- ☐ App unlock after backgrounding > `{{5 minutes}}`
- ☐ View sensitive data (payment info, full account number)
- ☐ Confirm high-value transactions (amount > `{{$100}}`)
- ☐ Change security settings

Fallback: PIN code → `{{expo-secure-store}}` stored hash (bcrypt, not reversible)

2.2 Session Management

Property	Value
Access token lifetime	`{{15 minutes}}`
Refresh token lifetime	`{{30 days}}`
Refresh token rotation	Yes — single use, new token issued on use
Session timeout (background)	App lock after `{{5 minutes}}` inactive
Max concurrent sessions	`{{3 devices}}`
Force logout triggers	Password change, suspicious activity, admin revocation

Token storage:

```
// CORRECT — encrypted secure storage
await SecureStore.setItemAsync('access_token', token, {
  keychainAccessible: SecureStore.WHEN_UNLOCKED_THIS_DEVICE_ONLY,
});

// WRONG — never use AsyncStorage or MMKV for tokens
// await AsyncStorage.setItem('access_token', token); // DO NOT USE
```

2.3 Token Storage (Secure Enclave)

Platform	Mechanism	Access Level
iOS	Keychain Services	`kSecAttrAccessibleWhenUnlockedThisDeviceOnly`
Android	Android Keystore	`BIOMETRIC_STRONG` or `DEVICE_CREDENTIAL`

Cannot be backed up: Both mechanisms are device-bound — tokens do not transfer to new devices.

3. Data Protection

3.1 Secure Storage Policy

Data Type	Allowed Storage	Forbidden Storage
Access token	Keychain/Keystore	AsyncStorage, MMKV, SQLite
Refresh token	Keychain/Keystore	AsyncStorage, MMKV
Encryption key	Keychain/Keystore	Any other storage
User PII	SQLite (encrypted)	AsyncStorage
Non-sensitive prefs	AsyncStorage / MMKV	—

3.2 Data Encryption at Rest

Database encryption: `{{SQLCipher for SQLite | Realm Encryption}}`

```
// SQLCipher initialization with key from Keystore
const encryptionKey = await SecureStore.getItemAsync('db_encryption_key');
const db = await SQLite.openDatabaseAsync('app.db', {
  key: encryptionKey,
});
```

Key generation: On first launch, generate 256-bit random key, store in Keychain/Keystore.

Encrypted fields (beyond DB encryption):

Field	Encryption	Key Source
Payment card (last 4 only stored)	N/A — tokenized	Stripe/Braintree
SSN / national ID	AES-256-GCM	Keychain/Keystore
Private messages	End-to-end (Signal protocol)	Derived keys

3.3 Sensitive Data in Memory

Rule	Implementation
Clear passwords after use	Overwrite string variable, trigger GC

Rule	Implementation
No sensitive data in logs	Custom logger strips PII fields
No sensitive data in crash reports	Sentry `beforeSend` hook scrubs payload
No sensitive data in analytics	Map user ID → hashed ID before sending

---

3.4 Screenshot Prevention

```
// iOS — prevent screenshots programmatically
// (Note: React Native does NOT expose this API — native module required)

// Android — prevent screenshots and screen recording
import { Platform } from 'react-native';
import { preventScreenCapture, allowScreenCapture } from 'expo-screen-capture';

useEffect(() => {
  if (Platform.OS === 'android') {
    preventScreenCapture();
  }
  return () => allowScreenCapture();
}, []);
```

Screens requiring screenshot prevention:

- ☐ Payment / card details screen
- ☐ Account balance screen
- ☐ Personal document viewer
- ☐ `{{OTHER_SENSITIVE_SCREEN}}`

3.5 Clipboard Protection

- ☐ Password fields: `secureTextEntry={true}` — disables clipboard by default on iOS
- ☐ Sensitive data programmatically copied: clear clipboard after `{{60 seconds}}`
- ☐ Custom clipboard hook: `useSensitiveCopy(value, clearAfterMs: 60000)`

---

4. Network Security

4.1 Certificate Pinning

Library: `{{react-native-ssl-pinning | OkHttp certificate pinner (Android native) | URLSession (iOS native)}}`

```
// Using react-native-ssl-pinning
const response = await fetch('https://api.domain.com/endpoint', {
  method: 'POST',
  pkPinning: true,
  sslPinning: {
    certs: ['api_cert_sha256_fingerprint'],
  },
  headers: { /* ... */ },
  body: JSON.stringify(payload),
});
```

Pinned endpoints:

- `api.{{domain.com}}` — primary API
- `auth.{{domain.com}}` — authentication

Certificate rotation process:

1. Pin BOTH current cert and backup cert simultaneously
2. Deploy app update with new cert added
3. After old cert expires, remove old cert from pins
4. Test rotation in staging first

Handling pin failures:

- Log as security event to backend
- Show user-facing error: "Secure connection failed"
- Do NOT fall back to unpinned connection

4.2 SSL/TLS Configuration

Requirement	Value
Minimum TLS version	TLS 1.2

Requirement	Value
Preferred TLS version	TLS 1.3
HTTP allowed	No (ATS enforced on iOS, enforce on Android)
Cipher suites	Forward-secrecy only (ECDHE, DHE)

4.3 Network Request Encryption

- ☐ All requests over HTTPS (enforced via ATS + Android network security config)
- ☐ Sensitive payloads encrypted at application layer (in addition to TLS): `{{Yes/No}}`
- ☐ Request signing implemented (HMAC): `{{Yes/No}}`

---

5. Application Security

5.1 Code Obfuscation

Platform	Tool	Status
Android	ProGuard / R8 (enabled by default in release)	`{{Done/TODO}}`
iOS	Bitcode + Swift compiler optimizations	`{{Done/TODO}}`
JavaScript	Hermes bytecode compilation	`{{Done/TODO}}`
JavaScript (extra)	`{{metro-react-native-babel-preset obfuscation}}`	`{{Done/TODO}}`

ProGuard rules: `android/app/proguard-rules.pro` **TODO:** Review ProGuard rules to ensure no over-stripping of reflection-dependent code.

5.2 Root / Jailbreak Detection

Library: `{{expo-device | react-native-jail-monkey | custom}}`

```
import JailMonkey from 'jail-monkey';

export function checkDeviceIntegrity(): SecurityResult {
  const isJailbroken = JailMonkey.isJailBroken();
```

```
  const isDebuggedBuild = JailMonkey.isDebuggedMode();
  const onExternalStorage = JailMonkey.isOnExternalStorage(); // Android

  return {
    compromised: isJailbroken || isDebuggedBuild,
    reason: isJailbroken ? 'jailbroken' : isDebuggedBuild ? 'debug' : 'clean',
  };
}
```

Response to compromised device:

- `{{Warn user and allow use with reduced functionality | Block access entirely | Log silently}}`
- Justification: `{{Explain decision}}`

5.3 Tamper Detection

- ☐ App bundle signature verification on launch
- ☐ Resource hash verification for critical assets
- ☐ Backend validates app version — block known compromised versions

---

5.4 Debug Detection

```
// Detect if app is running under debugger in production
if (__DEV__ === false && detectDebugger()) {
  logSecurityEvent('debugger_attached');
  terminateSession();
}
```

Production builds must have:

- ☐ `__DEV__` mode disabled
- ☐ Console.log stripped (Babel plugin: `transform-remove-console`)
- ☐ Flipper disabled
- ☐ Source maps NOT bundled with the app binary (upload to Sentry separately)

5.5 Reverse Engineering Prevention

Measure	Implementation
No hardcoded secrets	All secrets fetched from vault at runtime or injected via CI
No plain API keys in bundle	Obfuscated + fetched from secure config endpoint
Sensitive logic on server	Business logic requiring security stays backend-side
Binary protection	ProGuard / R8 (Android), Bitcode stripping (iOS)

TODO: Run MobSF static analysis before each major release and review findings.

---

6. OWASP Mobile Top 10 Checklist

#	Risk	Status	Notes
M1	Improper Credential Usage	`{{Pass/Fail/N/A}}`	Tokens in Keychain, no hardcoded creds
M2	Inadequate Supply Chain Security	`{{Pass/Fail}}`	Dependency audit `npm audit`
M3	Insecure Authentication/Authorization	`{{Pass/Fail}}`	JWT validated server-side
M4	Insufficient Input/Output Validation	`{{Pass/Fail}}`	Zod validation on all inputs
M5	Insecure Communication	`{{Pass/Fail}}`	TLS + cert pinning
M6	Inadequate Privacy Controls	`{{Pass/Fail}}`	Data minimization, GDPR
M7	Insufficient Binary Protections	`{{Pass/Fail}}`	Obfuscation, no debug build in prod
M8	Security Misconfiguration	`{{Pass/Fail}}`	No dev configs in prod build
M9	Insecure Data Storage	`{{Pass/Fail}}`	Keychain/Keystore + encrypted DB
M10	Insufficient Cryptography	`{{Pass/Fail}}`	AES-256, SHA-256, no MD5/SHA-1

---

7. Security Testing Tools

Tool	Type	When	Output
MobSF (Mobile Security Framework)	Static analysis	Pre-release	Risk report

Tool	Type	When	Output
Frida	Dynamic analysis	Penetration test	Runtime behavior
Objection	Runtime analysis	Penetration test	Bypass checks
Burp Suite	Network proxy	Penetration test	API vulnerabilities
OWASP ZAP	Automated scan	CI/CD	Vulnerability report
`npm audit` / `yarn audit`	Dependency scan	Every PR	CVE report

Penetration test cadence: `{{Annual | Before each major release | Quarterly}}` **Last pentest date:** `{{DATE}}` **Next pentest date:** `{{DATE}}`

---

8. Compliance Requirements

Requirement	Applicable	Notes
GDPR (EU users)	`{{Yes/No}}`	User data deletion, export endpoints
CCPA (California users)	`{{Yes/No}}`	Do Not Sell option
COPPA (under 13)	`{{Yes/No}}`	Parental consent flow
PCI DSS (payment data)	`{{Yes/No}}`	Use PCI-compliant SDK (Stripe/Braintree) — never store card data
App Store privacy label	Yes	App Store Connect privacy questionnaire
Play Store data safety	Yes	Google Play Console data safety form
HIPAA (health data)	`{{Yes/No}}`	BAA with vendors, encryption at rest+transit

---

Approval

Role	Name	Date	Signature
Author			
Mobile Lead			
Security Lead			
Legal			

Revision #7
Created 2026-02-23 12:03:08 UTC by John
Updated 2026-05-25 07:33:19 UTC by John