approov

CHECKLIST

Mobile App Scraping & API Abuse Assessment

The Situation

Mobile applications have become the primary interface between your business and its most valuable data—pricing, inventory, availability, user behavior, and proprietary signals.

Historically, organizations focused on web scraping and browser-based abuse. That model no longer reflects reality.

Today, sophisticated actors bypass mobile apps entirely by impersonating them. They reverse-engineer mobile applications (especially Android), replicate API calls, and extract structured data directly from backend systems—often at scale and without detection.

This shift has been accelerated by AI-driven automation, which dramatically lowers the cost and increases the effectiveness of data harvesting.

Why This Matters to Leadership

This is no longer a narrow security issue. It is a business, revenue, and competitive-advantage issue.

When mobile APIs are exposed:

- Pricing and inventory data becomes a commodity
- Demand and behavioral signals leak silently
- AI systems can be trained on your proprietary data
- Partner and distribution controls are undermined
- Abuse becomes harder to detect and easier to scale

Once data is extracted, it cannot be recalled. Legal remedies are slow, uncertain, and often ineffective after the fact.

How to Score Your App

- For each section:
  - Answer the assessment questions
  - Follow the how-to evaluate steps
  - Note findings and gaps
- Treat ❌ answers as current exposure, not theoretical risk
- Assume motivated attackers with modern tooling and AI assistance

---

1. Client Authenticity & Trust Model

Core question: Can your backend prove that an API request came from your genuine, untampered mobile app?

This is the single most important control.

What to assess

- ☐ Do API requests include proof of app authenticity (not just user identity)?
- ☐ Is the proof cryptographically verifiable server-side?
- ☐ Is the proof short-lived and non-replayable?
- ☐ Is access denied or restricted if proof is missing or invalid?
- ☐ Is app authenticity evaluated continuously, not just at install time?

How to evaluate (hands-on)

1. Replay test

- Capture a legitimate API request (e.g., with a proxy)
- Replay it using curl/Postman
- Observe: does the backend accept it?

2. Client substitution

- Remove the mobile app entirely
- Recreate requests in a script
- If successful backend trusts the protocol, not the app

3. Token reuse

- Reuse tokens from another device or environment

- If valid → tokens are not bound to app context

Common failure modes

- "We authenticate users, so we're safe"
- API keys embedded in apps
- Long-lived OAuth/JWT tokens
- Trust based on endpoint or User-Agent

Risk if weak: Any attacker who understands your API can access it at scale.

---

2. Reverse Engineering & App Cloning Resistance

Core question: How easy is it to understand and replicate your app's API behavior?

What to assess

- ☐ API endpoints and parameters are not trivially discoverable
- ☐ TLS interception is detected and handled
- ☐ Runtime hooking/instrumentation is detected
- ☐ Repackaged or modified apps are blocked
- ☐ Emulator/rooted environments are addressed appropriately

How to evaluate

1. Static analysis

- Decompile the APK/IPA (e.g., JADX)
- Can endpoints, headers, or secrets be identified?

2. Dynamic analysis

- Run the app behind a proxy (mitmproxy, Charles)
- Can traffic be intercepted without disruption?

3. Runtime tampering

- Inject a basic hook (e.g., Frida)
- Does the app continue to function?

4. Repackaging

- Modify the app slightly and re-sign
- Can it still connect to production APIs?

Risk if weak: Attackers can fully automate your app without using it.

3. API Surface Area & Data Sensitivity

Core question: Which APIs would cause damage if scraped or automated?

What to assess

For each API endpoint, classify:

- [ ] Data type (pricing, inventory, PII, content, analytics)
- [ ] Sensitivity (low / medium / high)
- [ ] Business impact if harvested
- [ ] Frequency and volume of access

How to evaluate

1. Inventory endpoints

- List all mobile-accessible APIs
- Identify those returning structured JSON

2. Threat modeling

- Ask: “What could someone do with this data at scale?”

3. Prioritization

- Rank endpoints by business impact, not convenience

High-risk categories

- Pricing and inventory
- Availability and search
- User profiles and content
- Reviews, ratings, rankings
- Partner and affiliate APIs
- Internal analytics or feature flags

4. Authentication vs Authorization Reality Check

Core question: Does authentication automatically grant access to sensitive APIs?

What to assess

- [ ] Authentication ≠ authorization
- [ ] Sensitive APIs require additional trust signals
- [ ] Backend policy considers client legitimacy
- [ ] Access decisions are contextual, not binary

How to evaluate

1. Identify an authenticated token

2. Attempt access from:

- Another device
- A script
- A modified client

3. Observe whether access differs

Risk if weak: Any valid token can be weaponized by automation.

5. Token, Session, and Replay Abuse

Core question: Can valid credentials be reused outside your app?

What to assess

- [ ] Tokens are short-lived
- [ ] Tokens are bound to app/device context
- [ ] Replay from other clients fails
- [ ] Session misuse is detectable

How to evaluate

1. Extract a token

2. Reuse it from:

- Another device
- A script
- An emulator

3. Measure how long it remains valid

Risk if weak: Attackers need to steal credentials only once.

---

6. Automation & Bot Detection Reality

Core question: Are you trying to detect bad behavior instead of preventing it?

What to assess

- [ ] Rate limiting is not the primary defense
- [ ] IP reputation is not relied upon
- [ ] Behavior-based detection is considered unreliable
- [ ] Controls assume AI-level attackers

How to evaluate

1. Simulate low-rate, distributed traffic

2. Mimic real user behavior patterns

3. Observe detection thresholds

Risk if weak: You are playing an endless cat-and-mouse game.

---

7. AI & Data Exploitation Risk

Core question: Would scraped data create long-term strategic harm?

What to assess

- [ ] Data could train AI models
- [ ] Data reveals internal strategy or signals
- [ ] Data cannot be revoked once scraped

- [ ] Data reuse would be invisible

How to evaluate

1. Could this data improve someone else's product?

2. Could it be aggregated over time?

3. Would we ever know?

---

8. Embedded Secrets & Third-Party Abuse

Core question: Are secrets embedded in the mobile app?

What to assess

- [ ] No API keys in app binaries
- [ ] Secrets delivered dynamically
- [ ] Secrets bound to legitimate app instances
- [ ] Keys are rotatable without redeploying apps

How to evaluate

1. Scan binaries for keys

2. Monitor third-party API usage

3. Correlate usage to legitimate app traffic

Risk if weak: Attackers may abuse *your own paid services*.

---

9. Monitoring, Telemetry & Visibility

Core question: Would you know if scraping is happening today?

What to assess

- [ ] Ability to distinguish real apps from scripts
- [ ] Visibility into client integrity failures
- [ ] Endpoint-level abuse metrics
- [ ] Correlation between mobile and backend signals

How to evaluate

1. **Look for "normal" traffic assumptions**
2. **Ask what evidence proves legitimacy**
3. **Review alerting and dashboards**

---

10. Incident Response & Control Agility

Core question: Can you stop abuse immediately without breaking users?

What to assess

- [ ] Ability to block illegitimate clients instantly
- [ ] Ability to protect only high-risk endpoints first
- [ ] No dependency on app store redeploys
- [ ] Fail-closed options exist

How to evaluate

Run a tabletop exercise:

- **"Scraping confirmed on endpoint X — what happens in 1 hour?"**

Exposure Scoring

Score	Interpretation
0–2 weak areas	Low exposure (rare)
3-5	Moderate exposure
6-8	High exposure
9-10	Structural risk

Most mobile platforms fall in the 6–8 range.

Controls That Matter

Marginal defenses

- Rate limiting

- IP blocking
- CAPTCHAs
- Legal terms

Structural defenses

- App attestation
- Cryptographic app authenticity proof
- Zero-trust mobile APIs
- Runtime secret delivery

Final Self Test

Can your backend mathematically prove that a request came from your genuine mobile app?

If not, assume scraping and automation are already possible.

If this assessment surfaced any uncertainty, gaps, or unanswered questions, the next step is expert validation.

Solving mobile API abuse requires provable mobile app authenticity, enforced at the API layer, and aligned to real attacker capabilities.

Approov's mobile security experts specialize in helping teams:

- Validate whether their backend can cryptographically prove app authenticity
- Identify which APIs are already exposed to scraping or automation
- Prioritize protections based on business impact, not theory
- Implement controls that stop abuse immediately - without app redeployments

Start the conversation now: https://approov.io/info/contact