

GUIDE

Protecting Mobile Apps Against Malware

A practical guide for security and engineering teams. What mobile malware actually does to your app, why server-side controls cannot see it, and how to build a defence on devices you do not own.



Contents

01	Why this is your problem, not the user's	03
----	--	----

02	The threat landscape	04
----	----------------------	----

03	How malware arrives, and what it does	06
----	---------------------------------------	----

04	Why your existing controls cannot see this	09
----	--	----

05	An end-to-end model: prevent, detect, respond, predict	10
----	--	----

06	Six features to demand	11
----	------------------------	----

07	Designing a response policy	12
----	-----------------------------	----

08	Building the programme & what to measure	13
----	--	----

09	Evaluation checklist	14
----	----------------------	----

10	Where SecIron fits & conclusion	15
----	---------------------------------	----



01

Why this is your problem, not the user's

Mobile malware runs on a customer's personal phone. You did not choose the device, you cannot patch it, you cannot scan it, and you will never see it. It is tempting to conclude that an infected phone is the user's problem.

It is not, for one reason: when malware on that phone moves money out of an account through your app, the loss, the regulatory exposure and the headline all belong to you. The customer will not describe it as a malware infection. They will describe it as your app being unsafe.



You cannot clean the device, so the app has to keep working safely on the assumption that the device may already be compromised.



02

The threat landscape

Banking trojans are now the dominant category of mobile malware. According to Kaspersky's Securelist mobile threat statistics for the first quarter of 2026, Trojan-Banker accounted for roughly 53 per cent of detected mobile malware, with more than 162,000 distinct banking trojan installation packages found in that quarter alone. Kaspersky also recorded a 196% rise in banking trojan attacks on smartphones during 2024, exceeding 1.24 million attacks on Android.

53%

of detected mobile malware in Q1 2026 was Trojan-Banker (Kaspersky Securelist)

196%

rise in banking trojan attacks on smartphones during 2024, exceeding 1.24m attacks on Android

162,000+

distinct banking trojan installation packages found in Q1 2026 alone

Research by Bitsight places this activity as especially prevalent in Brazil, Turkey and South East Asia, and notes that campaigns are customised for local financial institutions and payment platforms. This is not undirected malware that happens to reach your customers; overlays are built to match specific banks' login screens, in the local language.



The threat landscape, cont.

Two findings illustrate how targeted this has become. Cyfirma documented a trojan distributed inside counterfeit versions of Indonesia's digital national identity card app, which took real-time screenshots to map banking app layouts. Separately, researchers described a family named FjordPhantom that uses virtualisation and hooking to hide malicious accessibility services from detection.



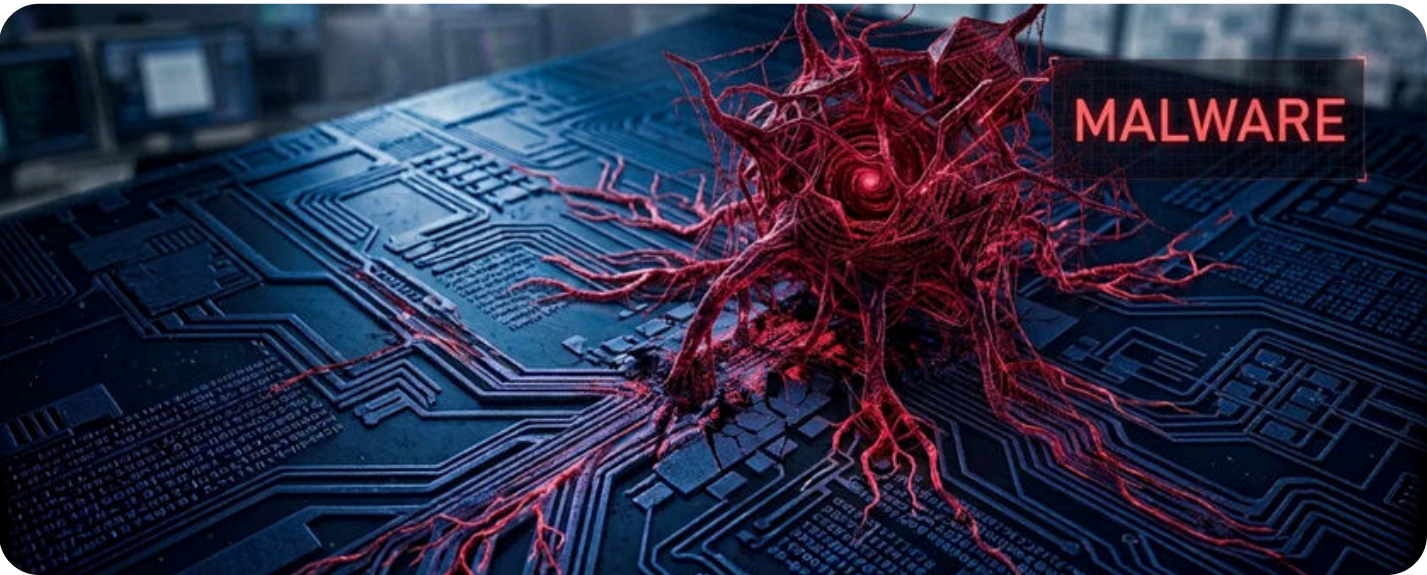
Nor does store review filter this out. A dropper carrying the Anatsa banking trojan was published on Google Play as a document viewer utility, reached 90,000 downloads and fourth place in its category, before activating its payload six weeks later. The app that passes review is not the app that attacks your customer.

Platform matters too. Attackers deliberately split their approach within the same campaign; phishing aimed at iOS, malware aimed at Android; so a single-platform defence leaves half the campaign untouched.



03

How malware arrives, and what it does



Six routes onto the device. Five of them are outside your control, which is the practical case for defending inside the app rather than trying to police the phone.

Route	How it works	In your control?
Droppers in official stores	A benign utility passes review, then fetches its payload weeks later	No
Smishing and phishing	A delivery, tax or bank message leads to a sideload prompt	Partly
Third-party stores	Users seeking free or region-locked builds install trojanised copies	No
Counterfeit apps	Malware ships inside a fake government, banking or utility app	Partly
Malicious advertising	An advert pushes a fake update or security tool	No
Compromised SDKs	A third-party library in your own app ships malicious code	Yes



The capability stack

Modern families are modular and combine several of these. Published analyses of the TrickMo family, for example, describe phishing overlays, keylogging, screen recording, live screen streaming, SMS interception and suppression of one-time password notifications in a single package.

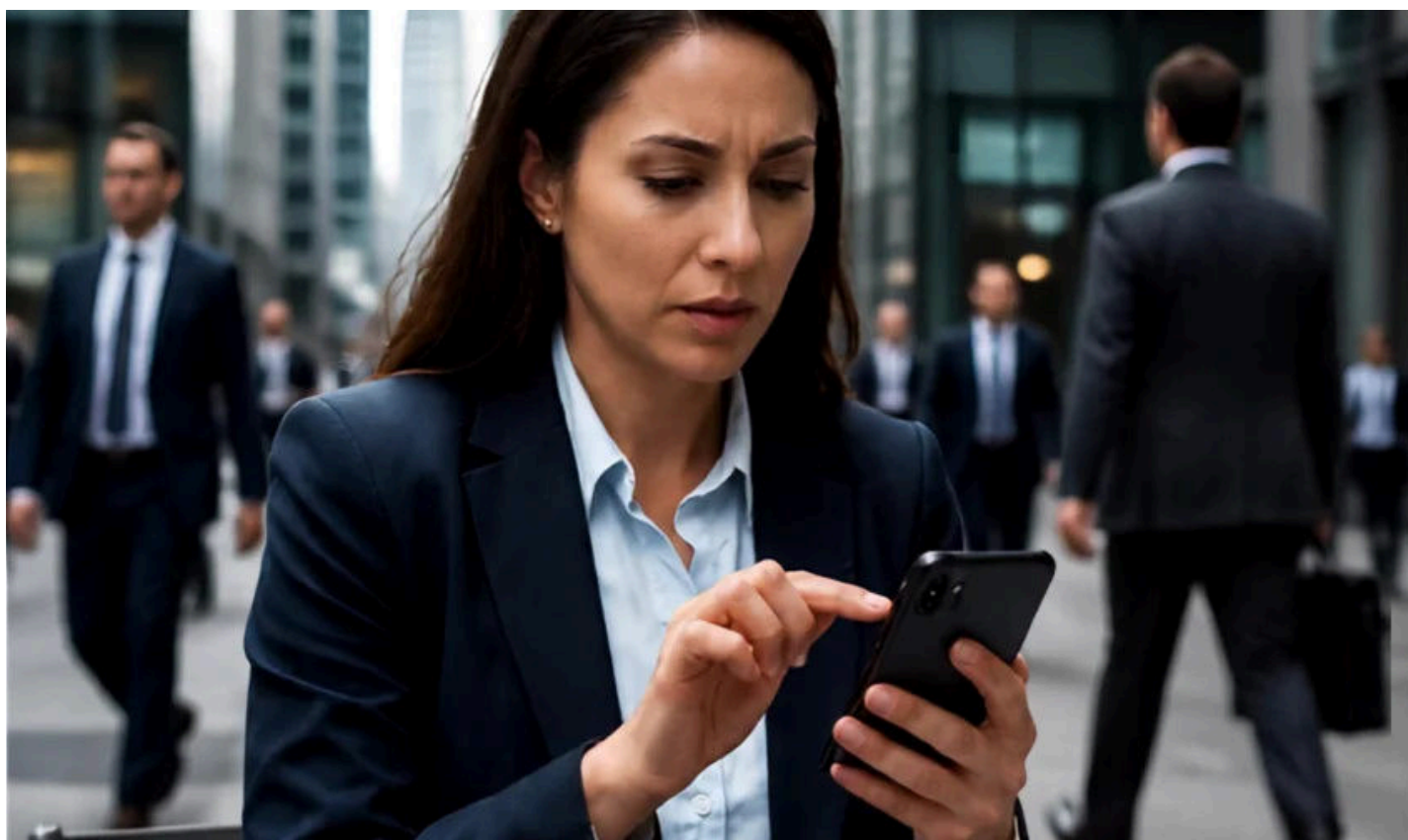
Capability	What happens	Why your app never sees it
Overlay attacks	A copy of your login screen is drawn on top of your app; credentials go to the attacker	Your app records no failed login; nothing happened to it
Accessibility abuse	Malware obtains the permission assistive software uses, then reads the screen and taps anything on it	The permission is granted legitimately by the user
Screen capture & streaming	Balances, statements and OTPs sent to the attacker; some map field positions for later automation	Reading the screen is an OS function, not an app event
Keylogging	Credentials, PINs and card numbers captured as typed	Input arrives at your app looking normal
SMS interception	One-time passwords read on arrival and their notifications hidden	Your authentication succeeds, correctly
On-device fraud	The attacker operates your app on the victim's phone, inside the victim's authenticated session	Every signal is genuine; see the next chapter
Virtualisation & hooking	The app is run inside a container or rewritten in memory, hiding the attack from checks	Nothing on disk changes, so integrity checks pass
Anti-removal logic	Uninstall attempts are detected and dismissed; revoked permissions re-grant themselves	Telling the customer to remove it may not work

That last row is documented by CybelAngel, whose investigation of Android banking trojans found anti-removal logic across several families, including automatic re-granting of the accessibility permission.



On-device fraud is the reason mobile malware defeats defences that work well elsewhere

As CybelAngel's researchers put it, the transaction originates from within the victim's own authenticated session; so it carries the correct device, the correct address, the correct app instance and a legitimate login.



Device fingerprinting, IP reputation and behavioural anomaly detection all pass, because every signal they read is genuine. The only thing wrong is who is holding the phone.



04

Why your existing controls cannot see this

Control	Why it misses mobile malware
API gateway and WAF	The request is well-formed and correctly authenticated. Nothing in it is anomalous
Fraud and risk engines	Device, location and session signals are genuine. Behavioural models catch some automation patterns, but not reliably
Multi-factor authentication	SMS codes are intercepted and their notifications suppressed. Push approvals are tapped by accessibility automation
App store review	Droppers activate weeks after approval, and clones are distributed outside the store entirely
Mobile device management	Applies to corporate devices. Your customers' phones are not enrolled and never will be
Consumer antivirus	Depends on the customer installing it, keeping it running, and the malware not evading it
Penetration testing	Tests your app's own flaws, not a hostile environment around an app that has none

None of these should be removed. The point is that none of them observe the device at the moment fraud occurs; and the app is the only component that can.

TIP This is the gap IronSky is built for: real-time monitoring on the device itself, so a zero-day technique is flagged and responded to the moment it appears, not discovered weeks later in a post-incident review.



05

An end-to-end model: prevent, detect, respond, predict

SecIron embraces the Adaptive Security Architecture (ASA) principles, which aim to Prevent, Detect, Respond and Predict threats.

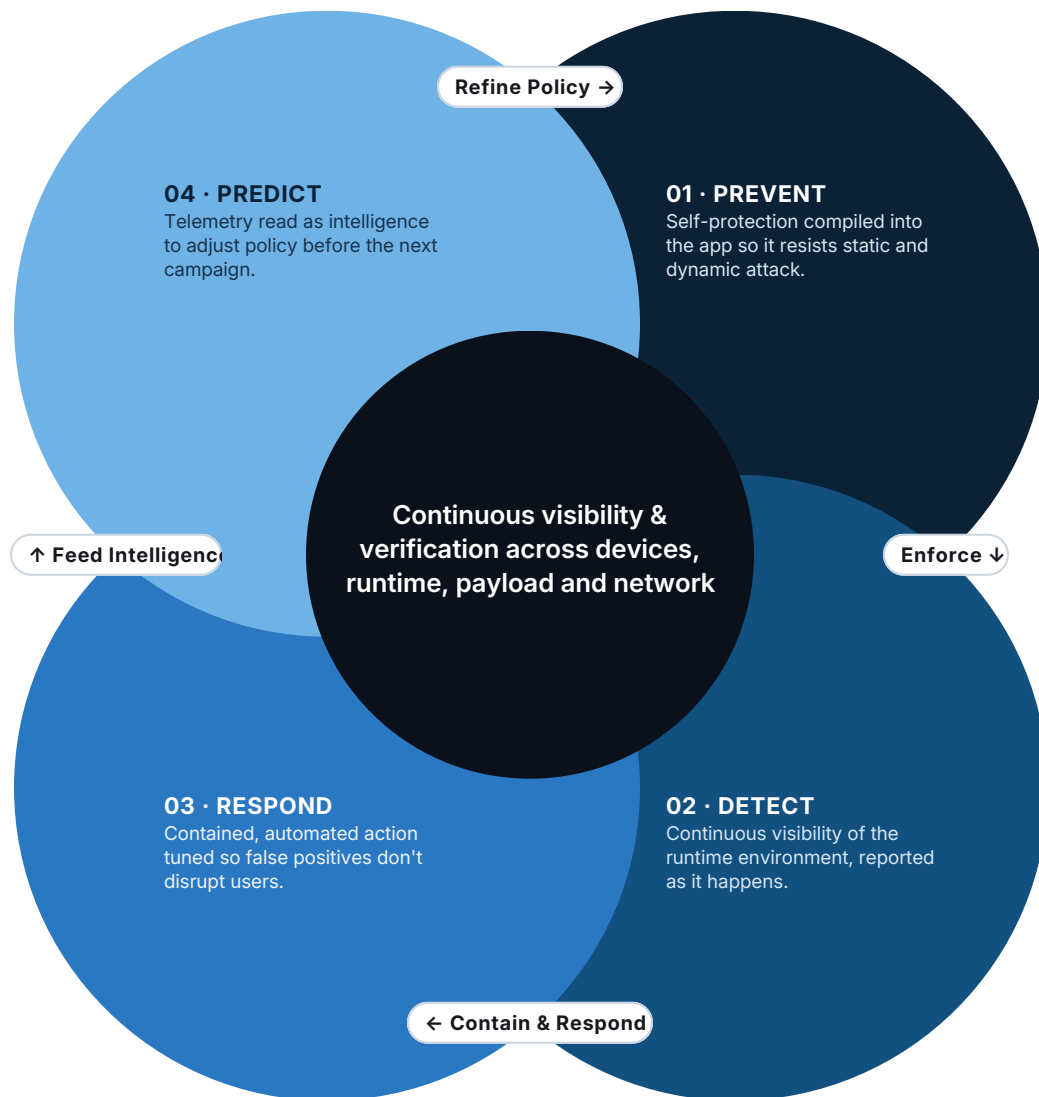


Figure: the ASA loop as applied to mobile; each stage feeds the next, so detections from live traffic continuously refine prevention and policy across the app portfolio.



Six features to demand

Whatever you shortlist, these six separate protection that holds from protection that reads well on a datasheet. Each is verifiable in a proof of concept rather than a document.

Feature	What it should mean	Why it matters
Anti-malware depth	Cloud-assisted threat intelligence, signature detection and behaviour-based detection working together	Signatures name known families quickly; behaviour catches the family that appeared last week
Codeless integration	Protection applied to the built app, with no source-code changes and no SDK work	Hardening that needs engineering time competes with the roadmap, and loses quarter after quarter
Flexible deployment	SaaS where agility matters, on-premises where regulation requires it	Regulated institutions cannot always send builds to a public cloud
Flexible policy enforcement	Policies set per threat and per app, by you rather than by vendor default	A blanket policy generates support tickets, and support tickets get protection switched off
Centralised management	One console covering every app in the portfolio	Per-app reporting hides the pattern that tells you a campaign is running
Real-time protection	Detection, response and reporting at the moment of the event, not in a nightly batch	On-device fraud completes in seconds

SecIron provides all six. The checklist later in this guide turns them into questions you can put to any vendor, including us.

TIP Not sure which solution meets all six? [Talk to our team of experts](#); we'll map your requirements against the checklist and show you where SecIron fits.



Designing a response policy

This is where deployments most often fail, and the failure is predictable. A blanket policy that ends the session on any detection produces support tickets, support tickets produce pressure, and within two quarters the control is off and nobody remembers deciding to turn it off. Design by severity and by what the user is doing at the time.

Detection	Suggested response	Reasoning
Accessibility service reading your app	Block the sensitive screen, warn clearly	Strong correlation with active fraud, but genuine assistive users exist; give them a path
Overlay over a payment screen	Suspend the transaction	Almost never legitimate at that moment
Known trojan signature present	Exit the session, notify the customer	Named, confirmed threat
Rooted or jailbroken device	Restrict high-value functions	Not malware by itself; blocking outright removes legitimate users
Screen recording during a transaction	Blank the sensitive fields	Proportionate, and preserves the rest of the session
Hooking or virtualisation	Exit the session	No legitimate reason for your app to run under instrumentation

Two principles worth writing down. Tell the customer something useful; a message naming what was found and what to do next turns a frightening moment into a trust-building one. And log every detection even where the response is to allow the session, because the log is how you learn what is happening in your own user base.



Building the programme

Phase	What you do
1. Baseline weeks 1–2	Inventory every mobile app, including those owned by other business units. Scan each for vulnerabilities, exposed secrets and risky third-party libraries; the SDK supply chain is the one route you fully control. Rank by what an attacker gains
2. Harden weeks 3–4	Apply protection to the highest-risk app. Turn detection on in monitoring mode with no enforcement. Verify against a real sample in a controlled environment rather than trusting the datasheet
3. Learn weeks 5–8	Watch the telemetry; you will find infection rates in your own user base, which is usually when the programme gets properly funded. Tune thresholds against real data. Draft customer messages and have support review them
4. Enforce week 9 on	Switch on enforcement, highest-severity detections first. Feed telemetry into your security operations centre. Add the hardening step to CI/CD so every release inherits it. Roll out across the portfolio

What to measure

 Track these five against your baseline every release cycle

- 1. Infection rate in your user base.** The share of sessions showing malware indicators; the number that sizes the problem.
- 2. Detections by type.** Which techniques are used against you specifically, which should drive policy.
- 3. Blocked transaction value.** The clearest return figure available, and the one that renews budget.
- 4. False-positive rate.** Track it deliberately. An unmeasured false-positive rate is what gets protection disabled.
- 5. Time from new technique to protection update.** Days is achievable. Weeks means weeks of exposure.



Evaluation checklist

Run this against any product you consider, ours included. Every item is demonstrable in a live session.

Ask	What a good answer looks like
Show me an overlay attack detected on a live device.	A demonstration, not a slide
How do you detect a malicious accessibility service?	A behavioural explanation of what is observed and why
What if the malware uses virtualisation or hooking to hide?	A specific mechanism; this is what current families do
Signature-based, behaviour-based, or both?	Both, with clarity about which catches what
Can we set different responses for different threats?	Per-threat and per-app policy, controlled by us
SaaS or on-premises?	Both, if regulation may require the second
What does our team see when a detection happens?	A live incident record with device conditions and action taken
Can we export it for an auditor?	A dated report readable without a product briefing
How fast do you ship protection for a new family?	Days, with a named process
How much source code do we change?	None. Protection applied to the built app
What is the performance cost?	A measured figure stated on the record, verifiable in a proof of concept



Where SecIron fits

SecIron covers the model above end to end, across three products.

IRON SCAN	Scans mobile apps for vulnerabilities and risky components; the baseline in Phase 1, including the third-party library supply chain
IRON WALL	Applies protection to the built app: rooting, jailbreaking, emulators, virtualisation, hooking, repackaging and malicious accessibility services detected through behaviour as well as signatures, with response policy set by you, and no source-code changes
IRON SKY	Monitoring and response. Every detection logged the second it happens, across the whole portfolio, exportable for audit

IronWall also enforces its own presence: attempt to strip the protection out of the binary and the app alerts, or exits; a capability no competitor fully matches, and one that matters because malware families now include logic to disable the defences they find.

About SecIron

Founded in 2010, SecIron is a leading provider of mobile application security solutions. Our platform; IronScan, IronWall and IronSky; combines vulnerability assessment, runtime hardening and real-time risk detection to deliver continuous, end-to-end mobile app protection for leading enterprises.

Years of experience

10+

Apps protected

10,000+

Customers globally

70+

Conclusion

Mobile malware is no longer a credential-theft problem that ends at your login screen. On-device fraud means the attacker uses your app, on your customer's phone, inside a session your systems correctly authenticated; and every signal your fraud stack reads will be genuine.

That leaves the app as the only place the fraud is observable while it happens. A defence that works assumes the device is already compromised, detects what is being done to the app rather than what the malware is called, responds proportionately enough that nobody disables it, and reports everything so your team can see the problem at all.

Talk to our team of experts to find out how Seciron can help secure your mobile apps.

[Talk to Us](#)

✉ business@seciron.com

🌐 www.seciron.com