PROMON

Promon Shield for SDKs™

Protect the SDKs powering your customers' apps.

Promon Shield for SDKs™

Shield for SDKs keeps your distributed SDKs secure, trusted, and tamper-resistant, wherever they're integrated. It prevents reverse engineering and unauthorized modification, strengthens runtime integrity, and makes adversarial and AI-assisted analysis significantly harder. With post-compile integration, you secure every SDK version without changing your development workflow.

Operating Systems	Android, iOS, macOS*, Google ChromeOS, Huawei HarmonyOS (excluding HarmonyOS Next), Amazon Fire OS
Languages	C, C++, Rust, Java, Kotlin, Objective-C, Swift, Dart
CI/CD Tools	GitHub Actions, Azure DevOps, Jenkins

*for mobile SDKs used in desktop environments

Why Promon Shield for SDKs™

Build trust across your partner ecosystem

Your SDK becomes part of your customers’ apps and security. Promon Shield for SDKs™ ensures your distributed code behaves securely wherever it runs, increasing partner confidence and reducing the risk of downstream compromise.

Protect your IP and competitive edge

Your SDK logic is valuable. Shield for SDKs applies multi-layer obfuscation, including VM-based obfuscation for Java-based SDKs, making your IP significantly harder to analyze or replicate, even with modern automated tools.

Security that doesn’t slow development down

Shield for SDKs integrates post-compile with no code changes. Developers keep shipping on schedule; DevSecOps teams automate protection across releases. You benefit from faster time-to-market, minimal developer friction, and consistent protection across every build.

Strengthen runtime integrity everywhere your SDK runs

SDKs often operate in environments you don’t control, including rooted, jailbroken, hooked, or emulated devices. Shield for SDKs detects and responds to these threats in real time to prevent malicious manipulations, as well as fraud and session abuse.

What sets Promon Shield for SDKs™ apart

Enhanced dynamic runtime protection

Detects rooting, jailbreaking, hooking, debugging, and emulation. Stops tampering before it causes harm.

Virtual machine obfuscation for Java

Executes selected Java methods inside a secure VM, making advanced or AI-assisted analysis significantly harder.

Layered code obfuscation

Section encryption, control-flow abstraction, string encryption, renaming, and block splitting protect your SDK structure and logic.

Broad language and platform coverage

Protect native, managed, and mixed SDKs across all major mobile platforms.

Post-compile integration

Apply security without modifying code or build systems.

How it works

Promon Shield for SDKs™ applies multiple layers of protection post-compile to secure your SDK's code, logic, and runtime behavior. These protections ensure your SDK remains tamper-resistant, harder to analyze, and more trustworthy across every partner integration. Shield for SDKs applies three complementary groups of defenses:

1 Obfuscation for native SDKs

Section encryption: Encrypts code and data sections within the native binary, preventing static analysis and hiding sensitive logic from attackers.

Control-flow abstraction: Reroutes calls to a central dispatcher, concealing the true control flow and hiding links between code blocks.

Checksumming & integrity checking: Adds a checksum network and per-function integrity verification. Unauthorized modification triggers unpredictable crashes or controlled shutdowns.

Block splitting: Breaks functions into small fragments and distributes them across the binary, making reverse engineering far more complex.

Objective-C renaming: Obfuscates class and method names to remove meaningful metadata from iOS binaries.

Debug stripping: Removes debug symbols and metadata that would otherwise aid attackers during analysis.

2 Obfuscation for Java bytecode

Debug attribute removal: Strips metadata to make decompilation harder.

String encryption: Encrypts sensitive static strings such as API keys, URLs, errors, and file paths.

Anti-decompilation: Alters bytecode metadata to break popular decompilers or force them to output invalid code.

VM obfuscation: Moves selected methods into a secure native virtual machine. Attackers cannot view real logic in the class files; they see only opaque stubs.

Control-flow obfuscation: Injects junk branches and misleading execution paths to conceal the SDK's true behavior.

Static member shuffling: Shifts static fields and methods between classes to break recognizable structures.

Renaming: Replaces class, field, and method names with meaningless identifiers and flattens package structure.

Constant hiding: Stores constants in separate classes or replaces them with computed values to obscure their purpose.

Debug detection: Checks during runtime for attached debuggers that may be attempting to analyze or manipulate the SDK.

App allowlisting: Ensures your SDK only runs inside approved apps, preventing unauthorized redistribution or abuse.

3 Dynamic runtime protections

Debugger detection: Identifies when a debugger attaches to the app, blocking dynamic analysis attempts.

Hooking detection: Detects injected code from hooking frameworks like Frida or LSPosed and responds immediately.

Emulator detection: Identifies when the SDK is running in an emulator, a common environment for automated attacks.

Root / Jailbreak detection: Detects compromised devices where system restrictions are removed, significantly reducing risk from malware or privilege-escalation attacks.

PROMON

About Promon

Promon leads the way in proactive mobile app security. For 19 years, we've been making the world a safer place by securing any app, on any device—in no time at all. Today, we protect over 2 billion users, secure 13 billion monthly transactions, and safeguard $2.5 trillion in market cap. Promon is headquartered in Oslo, Norway, with offices in more than 15 countries around the world.

Would you like to talk to an expert?

Mobile app security is crucial to preserve and improve your business reputation. Request pricing or talk to an expert to learn more today.

Book a meeting »