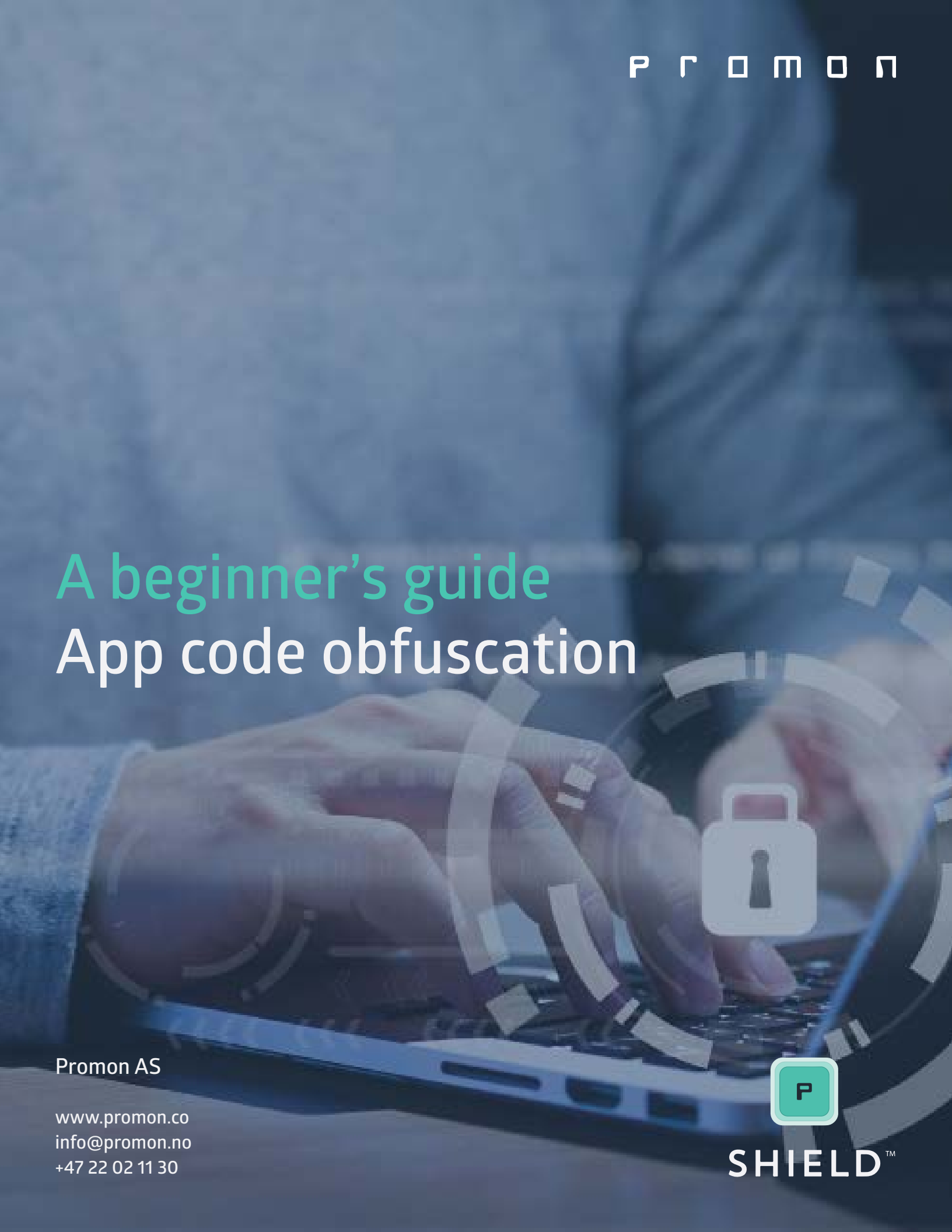
PROMON

A beginner's guide
App code obfuscation



Promon AS

www.promon.co
info@promon.no
+47 22 02 11 30

SHIELD™

Contents

Introduction

Mobile app-based cybercrime is ever-evolving, and hackers always find new and better methods to decompile and reverse engineer apps to identify weaknesses, secrets and get a hold of sensitive information.

Code obfuscation has become a standard method used by developers to prevent cybercriminals from decompiling and reverse engineering app code, protecting apps against intellectual property theft, loss of revenue, and reputation damage.

In this beginner's guide to obfuscation, you will get an introduction to what code obfuscation is, and a few different essential obfuscation techniques you should consider. You will also learn why it is important to use on apps for Android, iOS and JavaScript-based applications, and why you should combine obfuscation with multi-layered runtime protection.

What is code obfuscation?

At its core, obfuscation describes the act of obscuring or making something harder to understand. Thus, code obfuscation is a method of modifying an app's code to make it difficult for attackers to read and understand.

Obfuscation will help conceal the logic and purpose of an app's code, while still keeping its functionality. By using obfuscation methods, it will be more difficult for an attacker to perform reverse engineering, analyze the code, and retrieve sensitive information.

> **Code obfuscation makes it more difficult to perform reverse engineering, and to retrieve sensitive information.**

Advantages of code obfuscation

The threshold for an attacker to carry out a reverse engineer attack is significantly heightened if your app's code is obfuscated, as it will often be too costly and time-consuming to succeed in their attempt.

By implementing obfuscation methods and making the app hard to reverse engineer, you will more sufficiently protect against intellectual property theft and unauthorized access.

Prevent code from being **copied** and used without permission

Make your app's functional logic on the client-side and algorithms **less exposed**

Make it harder for attackers to find **vulnerabilities** in your code

Examples of obfuscation techniques

Renaming

This technique will rename for example classes and method names in a randomized fashion. It is often used to obfuscate application code developed in Java, .NET and Android platforms.

String obfuscation

String obfuscation hides string by scrambling them in various ways. It hides and replaces strings with a, for the human eye, nonsensical representation of the string. The CPU will be able to de-scramble the string dynamically during execution, while an attacker would make little meaning out of the string from static analysis.

Control flow obfuscation

When determining the intent of an application, it's important to understand its control flow. Control flow obfuscation is to alter the logical execution flow of the application by controlling the flow of application dynamic in a different and controlled manner. By complicating the control flow, by for example control flow flattening, it takes a lot longer to understand where the code is going or why it takes a certain direction.

Obfuscating iOS apps

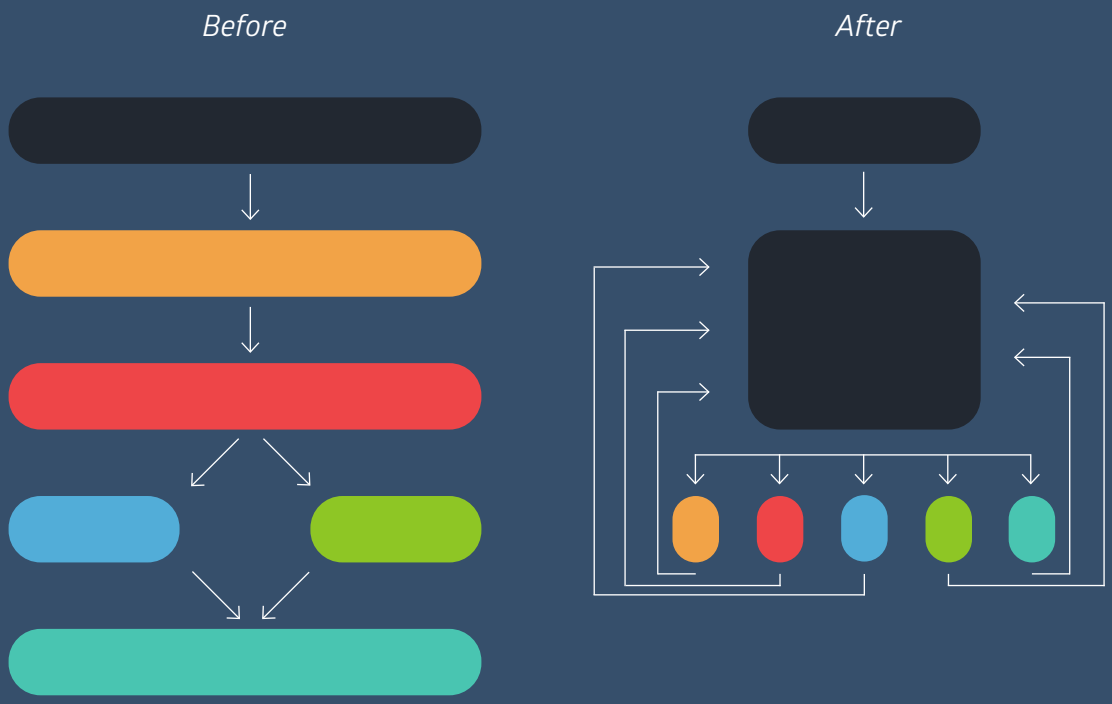
Although both Android and iOS apps are a constant target for hacking and reverse engineering, many think that iOS apps are not as exposed to such attacks as apps for Android.

Objective-C and Swift are the most common programming languages for iOS apps. Both are compiled to machine code, which makes it more difficult to translate the code back to the original source code. This has created a thought that iOS apps are hard to reverse engineer. However, the interest in analyzing machine code is nothing new, and there is a mature technology in place for reverse engineering machine code. In addition, when Objective-C and Swift are compiled to machine code, there is a lot of metadata in the binary that is required for these languages - which makes it much easier to understand the code than if you would for example write C code.

iOS apps are in other words not hard to reverse engineer and analyze, and it is important that you take the necessary steps to prevent these threats. You can obfuscate the code of your iOS apps with techniques such as control flow obfuscation and string obfuscation.

Control flow obfuscation

This illustration shows control flow flattening. By complicating the control flow, it takes a lot longer to understand where the code is going, or why it takes a certain direction.



Bitcode obfuscation

Bitcode is the intermediate representation of your source code in the process of compiling it and converting it into machine code. The front end of the compiler will convert the source code into Bitcode. Based on the optimized Bitcode, the compiler back-end generates machine code. Our in-app protection solution can work with the embedded Bitcode and apply obfuscation to it. It can then also compile it to binary format.

Obfuscating Android apps

The Android operating system is open source, which is an advantage for developers to be innovative and create groundbreaking apps, but this also leaves the apps prone to attacks from potential hackers.

Code obfuscation is the standard method to prevent hackers from decompiling and reverse engineering an app's code - but many Android apps don't have a sufficient level of protection, and often limit their obfuscation methods to code minification alone.

Unprotected Android apps increase the risk of exposing your businesses to IP theft, loss of revenue, or reputation damage. Android developers should choose a security software that applies advanced and multiple obfuscation techniques.

Obfuscation techniques for Android include renaming of class, function and method names, namespace flattening, and code shuffling.

Obfuscating JavaScript

JavaScript is currently the most popular programming language and has the highest number of contributors and repositories, outpacing other alternative programming frameworks. And it's more important than ever to consider JavaScript obfuscation.

Developing a single hybrid app is quicker and may be more cost-effective than developing native Android and iOS individually. However, hybrid apps can be more vulnerable to attacks than apps written using native languages because JavaScript is easier to reverse engineer and modify as it is not compiled into a more abstract representation in the published app.

The main objective of obfuscating your JavaScript code is to hide the parts of the code that attackers could target by hiding things like **strings, objects, and variables.** Essentially, obfuscation makes it hard to analyze your code to conceal the meaning of the data. In general, JavaScript obfuscation revolves predominantly around adding **entropy** – or complexity – to the JavaScript, to make it more difficult to understand.

Is code obfuscation alone enough?

Code obfuscation is a very effective method, as it will force an attacker to spend more time, effort and resources reverse engineering an app to visualise and understand its logic - and it should be implemented as part of your security.

But while obfuscation makes your business less prone to reverse engineering and intellectual property theft, you should not solely rely upon obfuscation as it does not protect your apps from malware or real-world attack scenarios.

Therefore, complete code protection combined with comprehensive runtime protection is essential to fully protect your apps. Choose a security product that applies advanced and strong obfuscation techniques in addition to other protection mechanisms.

Combine code obfuscation with runtime protection

Utilizing code obfuscation combined with multi-layered runtime protection features will make your apps less prone to reverse engineering and intellectual property theft.

Promon SHIELD™ offers the most comprehensive in-app protection solution on the market. In addition to applying robust obfuscation to your Android, iOS, and Javascript apps, Promon SHIELD™ will monitor your app's runtime behaviour and detect if your app executes in an insecure environment, e.g. hooking frameworks, emulators and debuggers.

Promon SHIELD™ also detects the presence of code hooks, blocks injection of malicious code into the app, and enables your app to modify its behavior in real-time to interrupt potential malware attacks.

SHIELD™

Would you like to talk to an expert?

In-App Protection is crucial to preserve and improve your business reputation. Promon's In-App Protection software Promon SHIELD™ obfuscates your app's code, making it significantly more difficult for an attacker to analyze. Request pricing or talk to an expert to learn more today.

Free demo

Promon AS

www.promon.co
info@promon.no
+47 22 02 11 30

SHIELD™