EndCrypt™ for Mobile

EndCrypt™ for Mobile

Table of Contents

Introduction

Introduction

EndCrypt is a white-box technology-based mobile application security solution that can easily turn your mobile applications into a self-protecting application within hours! It is a ready-to-use SDK (software development kit) integrated directly into the application instead of the network or operating system.

EndCrypt not only dynamically **Protects** your apps but also **Detects** and **Reacts** against zero-day and other wide-ranging cyber-attacks (malware, data leakage, intrusion, tampering, reverse engineering, etc.).

Along with (optional) **Manage** and **HSM Integratio**n capabilities, EndCrypt brought a new perspective and differentiate itself in the marketplace. HSM-integrated in-app security solutions ensure apps run securely more than ever!

Protect

2. Protection Capabilities (Hardening)

2.1. Code Obfuscation

Code obfuscation is one of the most important techniques that used to prevent reverse engineering attempt. The term "obfuscate" means to make obscure, unclear, hide or unintelligible. Along with this technique while the functionality of the code remains unchanged, the logic and the purpose of the app's code concealed.

2.1.1. Why Use a Code Obfuscator?

Code obfuscation is a process of modifying an executable code so that it becomes much more difficult for an attacker to understand how an application works.

For an attacker, the first step of gaining an understanding of app logic starts with a alysis of the source code. Therefore, making the source code as confusing as possible is vital for protecting apps agains curity threats, unauthorized access, and intellectual property theft.

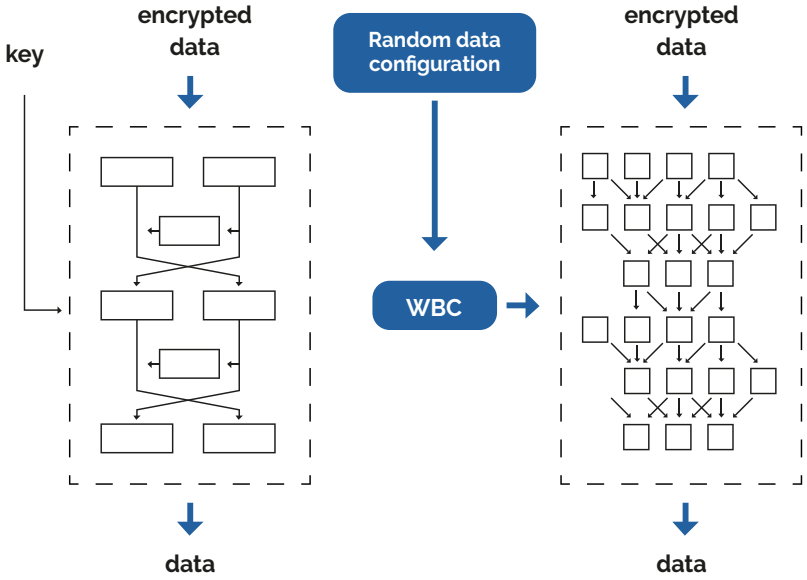
2.1.2. Should I Obfuscate My Application?

If you release a mobile app that includes intellectual property, offers access to sensitive information and operates in an untrusted environment, you should strongly consider using obfuscation methods.

Attackers will have a considerably harder time deciphering the code and analyzing the program if it is obfuscated, which also makes reverse-engineering difficult and economically unfeasible.

2.2. White-Box Cryptography

Today's applications contain a growing proportion of sensitive data, making them appealing targets for attack. At the same time, they are installed on endpoints that lay outside the scope of security controls. For this reason, developers must ensure that they have programs to safeguard cryptographic operations and keys wherever they execute. White-box cryptography, which is a software-based technology, provides security without the necessity for hardware support.



According to Gartner, white-box cryptography refers to the set of techniques used to hide and protect sensitive application data such as keys and credentials stored on a device. In its most basic form, white-box cryptography uses techniques similar to obfuscation to hide data, but it can also combine anti-tampering functionality. While code obfuscation modifies code to make it more difficult to decipher, white-box cryptography does more than that; it utilizes additional data transformation techniques designed particularly to protect software implementations of cryptographic algorithms. As a consequence, the secret cryptographic keys are always safe and hidden.

White-box cryptography and obfuscation are the main prevention capabilities (also refers to application hardening capabilities) that embed secret keys within application code. The aim is to mix code and keys in such a way that an attacker is unable determine the difference, thus the new white-box program can be run securely from an unsecured environment.

2.3. Resource Encryption

Resource Encryption is the other code hardening, prevention capability that ensures that the program's code and data cannot be viewed while the application is at rest. The primary objective is to ensure that keys and certificates are stored securely in the application, which usually involves cryptographic operations. When the application is executed, encrypted code is decrypted on the fly.

Encryption must be implemented across several layers to be successful. Some of the major encryption techniques are string encryption, class encryption, asset encryption, and resource encryption.
EndCrypt provides an encrypted secure environment on both Android and iOS that protects the integrity of keys and certificates in the application.
Technically, mobile application uses strong (e.g. NIST approved) encryption algorithms (RSA 2048, ECC P256 and AES 256).

2.4. Device Binding and Fingerprinting

Device binding or fingerprinting is the process used to identify a device based on its specific and unique configuration. According to Gartner, fingerprinting is a method that collects information about the device. The information can be used to uniquely identify a device, which allows the app to lock itself and only runs on that specific device (i.e., device binding).

2.5. Reverse Engineering

Hardening Applications Against Reverse Engineering
Reverse engineering is the process of analyzing an app to get a deeper insight about its original source code. This approach is widely used by attackers to study source code and create malware to exploit apps.

Applications published on commercial app stores are generally targeted by reverse engineering attacks. Attackers reverse engineer applications to figure out how it works and how they may be exploited in order to steal and clone data .
EndCrypt application hardening methods, such as code obfuscation, white-box cryptography, resource encryption make it more difficult for attackers to reverse engineer an application.

2.5.1. How to Protect Your Apps from Reverse Engineering?

Obfuscation tools such as string obfuscation, name obfuscation, control flow obfuscation, and arithmetic obfuscation are some of the most important tools that must be used to effectively protect your applications from reverse engineering attacks.

According to OWASP a strong obfuscator will be able to:

- Determine which procedures or code fragments to obfuscate
- Adjust the degree of obfuscation to balance performance impact
- Withstand de-obfuscation by tools such as IDA Pro and Hopper
- Obfuscate string tables as well as procedures

Along with code obfuscation techniques, EndCrypt also monitors runtime behavior and detects whether an app is running in an insecure environment. Therefore, by detecting hooks and blocking code injections, EndCrypt protects your app from the following:

- Intellectual property thefts
- Reputational damages
- Identity thefts
- Compromise of backend systems

2.5.2. Reverse Engineering Techniques

Attackers generally download the targeted apps from an app store and analyze them using a variety of tools in their local environment. (OWASP)

The most common attack scenarios in reverse engineering are;

- String Table Analysis
- Cross-Functional Analysis
- Source Code Analysis

2.6. Multi-Factor Authentication

Multi-Factor Authentication (MFA) is an access management component where users confirm their identity using at least two separate verification factors before accessing to a mobile app, website, or other online resource. If an attacker overcomes the first verification factor, there still has at least one more obstacle to overcome before gaining access to the target's account.

That is why there is a password and PIN that are created while signing up on EndCrypt. Thereafter these PINs and passwords are used for logging into the application. Also, there is a Soft OTP for authentication between the client and backend.

2.7. Man-in-the-Middle (MITM) Attack

Man-in-the-Middle (MiTM) is a type of cyberattack where the attacker silently relays and places himself or herself between two users to overhear their conversation or interrupt data transmission. An attacker is a silent observer and manipulator that intercept communication and message exchanges between two parties.

A session ID is generated each time a user interacts with a server/website. The main aim of the MiTM attack is to secretly capture this session ID in order to eavesdrop and manipulate the content. The ultimate purpose of this attack is to steal sensitive data such as login credentials, personal information, financial data, and etc.

Some types of MiTM attacks are; Email Hijacking, Session Hijacking, Wi-Fi Eavesdropping, IP Spoofing, DNS Spoofing, HTTPS Spoofing, SSL Hijacking, Browser Cookie Theft, ARP Spoofing, Rogue Access Point, mDNS Spoofing.

While trying to establish a connection with a server, apps don't usually determine which certificates to trust and which not to trust. Rather than accepting any certificate from a specific range of certification authorities, pinning allows the parties involved in the mutual authentication process to pin down particular certificates — only these certificates will be accepted. If an attacker spoofs a certificate, even if this certificate is coming from a legitimate certification authority, the communicating party will reject it, avoiding a man-in-the-middle attack. EndCrypt makes sure the protected application or SDK is connecting to the intended servers using SSL pinning, preventing man-in-the-middle attacks.

2.8. Runtime App Self Protection (RASP)

It is a security technology added to an application's runtime environment via code instrumentation. It monitors behaviors, controls application execution, and detects and prevents real-time attacks. This process consists of three independent parts.

1. At each start-up:

When the device starts, RASP values are taken, if any negativity is detected in any of the RASP controls, the everything sensitive is deleted.

Root Check - Hook Check - Debug Check - Emulator Check - CPU Check - API Check - Location Check

2. At MPA is running background:

This process runs continuously in the background every one second; everything sensitive is deleted if one of the cases below is detected.

Root Check - Hook Check - Debug Check - Emulator Check

3. At MPA is not running:

In case MPA does not run, all RASP controlled by background thread and all RASP information is sent to the backed system. This process runs continuously in the background with dynamically received params data from the backend.

Root Detect - Hook Detect - Debug Detect - Emulator Detect - CPU Check - API Check

2.9. Clickjacking Protection

According to OWASP, Clickjacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top-level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

Clickjacking protection includes methods such as;

- CSP
- SRI
- Script Injection Detection

According to Gartner's strategic planning assumption, by 2022, at least 50% of successful attacks against clickjacking and mobile apps could have been prevented using in-app protection.

- Complete Transparent Overlay
- Cropping
- Hidden Overlay
- Click Event Dropping
- Rapid Content Replacement
- Scrolling
- Repositioning
- Drag-And-Drop

2.10. Code Injection

Code injection is the term used to describe attacks that inject code into an application. Endcrypt enables your application or SDK to detect illegitimate code modifications and to verify the integrity of individual files with Tamper Detection.

Native C and JAVA programming languages are used to prevent code injection.

Detect

3. Detection Capabilities (Anti-tampering)

3.1. Emulation Detection

Emulators are generally used in reverse engineering attempts to examine a program to see how it works as well as to extract sensitive information that is exposed while the application is running. An emulator 'emulates' the whole system, including the processor in software. Since there is no official API for detecting emulators, many of the controls are done by the RASP system, which is the hardest challenge.

EndCrypt can identify whether a program is running in an emulator or a normal virtual environment.

3.1.1. Often-Used Android Emulators

- Android
- Andy
- ARChon
- Bluestacks
- Droid4X
- KoPlayer
- Droid4X
- Genymotion
- MEmu
- Nox
- PrimeOS
- emix OS Player
- Xamarin
- YouWave
- ArcWelder
- DroidDolphin

3.1.2. Commonly-Used iOS Emulators

- Smartface
- iPadian
- MobiOne Studios
- Air iPhone
- Appetize.io
- App.io
- XCode
- Remoted iOS simulator

3.2. Debugging Detection

Similar to emulators, debuggers are also used in reverse engineering attempts. It is commonly used during the run-time of the application to collect sensitive information. Debugging allows for a better knowledge of the malware's behavior, processes, and capabilities if executed properly. That is why attackers would want to prevent it for obvious reasons.

Anti-debugging techniques are designed to guarantee that a program is not running in a debugger and, if it is, to alter its behavior accordingly. In most situations, the Anti-Debugging procedure will slow down, but not stop, the reverse engineering process.

Anti-debug checks are enabled to detect if the application is running in debug mode and disable EndCrypt when a threat is detected.

3.3. Privilege Escalation Detection "Jailbreak or Root Detection"

Privilege escalation detection to detect if the device is jailbroken or rooted.

3.3.1. Root Detection (Android)

The term "Rooting", considered extremely insecure, is the process of bypassing the operating system's security measures in an Android application. Android apps run in isolated environments called Sandboxes, and attackers root a device to bypass an Android app's sandbox.

Since rooted Android devices users have almost complete control over the device and the data it stores, it poses a security risk to unauthorized access to sensitive data. Therefore, access to restricted data can be obtained without the need for authorization.

An application should run in a secure environment. A secure application provides an authentication mechanism that is critical to granting user access to a resource on the server. Therefore, in addition to recognizing a rooted device, app developers should make sure that their apps respond to capability.

EndCrypt detects the existence of a rooted device and responds by either notifying the user or terminating the program.

3.3.2. Jailbreak Detection (iOS)

Similar to rooting, jailbreaking, considered extremely insecure, is the process of bypassing the operating system's security measures in an iOS application established by Apple. This is generally done by device users in order to modify the device, alter or replace system applications and settings beyond what the manufacturer permits.

It's critical to secure your apps with solutions that detect vulnerable conditions and respond automatically in order to protect against malware and other risks posed by jailbroken devices.

Jailbroken device is considerably more vulnerable to being hacked, therefore it is very crucial to be aware of it. Leaving the app vulnerable to a jailbroken device will result in stolen sensitive customer data, loss of brand reputation and customer trust. EndCrypt detects the existence of a jailbroken device and responds by either notifying the user or terminating the program.

3.4. Malicious Software Check or Overlay Detection

On EndCrypt, there are two ways to check malicious software. If an application is added to this blacklist, during registration open session request this blacklist is checked, and a Tamper notification is sent to the device after in case of a match. It set the device status as disabled.

Secondly, if there is no match to the blacklist of malicious software, a permission score mechanism is applied for each software on the device.

3.5. Repackaging Detection or Cloning of the Device

Application repackaging is a technique, where unauthorized attackers download the app and change it to their benefit, add malicious functionality such as a keylogger to it, and upload it for illicit use by unwary users, who believe that they are using the original application. As a consequence, a fake version of a repackaged application is created.

An attacker may participate in application repackaging for a variety of reasons. This sort of attack may be used to install malware on a mobile device, posing a variety of business security threats. An attacker may add harmful functionality to an app and then redistribute it to users to penetrate networks, steal data, steal content, and eventually profit off the development of someone else's app.

EndCrypt allows you to identify and respond when an application is repackaged.

3.6. Anti-Tampering

The purpose of tampering mechanisms is to prevent an attacker from attempting an unauthorized physical/electronic action against the device.

EndCrypt uses a custom native library for all critical operations related to cryptographic calculations. EndCrypt checks checksum of the native library during registration, in each transaction time, in each session request. And if any chance is detected, then Local Database Encrypted (LDE) and White box library is removed.

EndCrypt have flow check mechanisms and double check mechanisms for critical path. And if any chance detected then Local Database Encrypted (LDE) and white-box library is removed.

3.7. Anti-Bot

Technologies are used to identify and/or block malicious bots based on behavior. CPU usage, RAM usage, and current device location are checked in each time on EndCrypt.

React

4. Reaction Capabilities (Pro-active Response)

4.1. Anti-screenshot Check

Prevent malicious and unintentional screenshots.

The screenshot is one of the easy methods to extract information from an application, where attackers use the screenshot to steal sensitive data and user credentials that are often displayed in apps. Therefore, mobile app developers should take necessary precautions and ensure that these data should not be exfiltrated.

This feature prevents to be screenshotted, and it totally disables this functionality to protect sensitive user data and the system. Screen capture and video capture functions are blocked on Create PIN and Change PIN screens on EndCrypt.

This function can also help avoid copyright theft and intellectual property violations.

4.2. Data Integrity Check

EndCrypt with runtime protection ensures the integrity of mobile apps in three ways: Protect, Detect and React, which fully protects sensitive business and personal data from cybercriminals. EndCrypt automatically verifies the structure and integrity of the app.

All sub-libraries are hashed and sent to a backend system. The backend system checks these files in each time.

4.3. Anti-Keylogging

Keylogger is one of the scariest spyware available today.

The term “keylogger” is the name given to software that monitors and tracks the activities on the keyboard. In fact, the basis of such software is to secretly monitor the keys on the keyboard. In this way, the attackers behind this monitoring software can see what is typed on the keyboard and secretly track the movements.

Attackers use this method to maliciously gain access to private information, credentials, credit card information etc., which results in loss of brand reputation and customer trust. The majority of keyloggers on iOS are implemented using code injection. EndCrypt’s code injection prevention feature protects against this type of attack. EndCrypt also provides a custom secure Procenne keyboard with randomly distributed numbers will be shown.

4.4. Risk Analysis – Risk Telemetry

It collects the so-called “attack telemetry” it has gathered from the application to process it in a backend system and decide how to react to a given level of risk. On EndCrypt, CPU, RAM, application list, location checks are done.

EndCrypt analyzes hundreds of telemetry data points gathered from iOS and Android devices and it offers app telemetry data for your mobile fleet in an easy-to-query format.

Manage

5. Manage Capabilities (Application Management & HSM Integration)

5.1. HSM Integration

ProCrypt HSMs in which encryption keys are securely stored and managed while cryptographic functions are executed. EndCrypt Backend and HSMs constitute "Procenne Digital Security Platform" serving as an end-to-end application security backbone system.

5.2. Secure Key Storage

White-box mechanism ensures the keys are stored safely and also on the server-side, as HSM does.

5.3. Secure Channel

This is the SDK component that manages the initialization and utilization of a "Secure" channel between the Mobile Devices and the EndCrypt Backend Application for exchanges of security keys and session-based security tokens. We use TLS 1.2 or above.

5.4. Stored Sensitive Data Protection

Application stores single-use keys and unique secure channel keys encrypted under White box Key in LDE. All stored data is encrypted as device bound. No key is stored in memory.
Keys are retained in memory only for the required duration. All are securely wiped just after usage.

5.5. Temporary Sensitive Data Protection

Temporary sensitive data on session should be used for operations after that it should be properly disposed of, not to be stored such as temporary keys.

5.6. Attestation and Monitoring

This component is responsible for monitoring the system health and operational status, such as CPU usage, RAM usage, current device location.

File, root, hook, emulator, debug check, API controls are monitoring also, blocking devices will be shown on the backend system.

5.7. Encrypted Database

This component is where data integral to the operation of the SDK components on the Mobile Device are stored and accessed securely.

5.8. Installation Certificate Check and Installation Source Check

EndCrypt gives an application the ability to ensure it has been signed with the original certificate and to check source control.

5.9. Secure Libraries Usage

EndCrypt enables your application or SDK to detect illegitimate code modifications and to verify the integrity of individual files. In addition, it is used for interlibrary communication masking methods to cover parameters.

5.10. Key Management System

"ProCrypt HSM" modules in which encryption keys are securely stored and managed, and cryptographic functions are executed. Designed and manufactured by Procenne, "ProCrypt HSM" is a Hardware Security Module (HSM) to be used for secure generation, storage, and distribution of cryptographic keys as well as for very fast processing of cryptographic algorithms.

About Us

Procenne Works for the Best!

Procenne produces "**Simple, Stable and Secure"** products and solutions for mobile application security, e-mail security, database and file security, KVKK (GDPR) infrastructure security, SSL termination and digital transformation projects in the field of digital security.

Established in 2013 headquartered in Istanbul, Procenne manufactures high-tech products that are critical in the digital security market. Procenne, which is also an R&D center, is the manufacturer of Turkey's first commercial Hardware Security Module, ProCrypt HSM device.

ProCrypt is a hardware security module that performs key management operations such as generating, storing and spreading keys used in encryption processes at high speed. The product, whose R&D studies started in 2013, became a commercial product in 2018 and received its international certificates, the **Common Criteria EAL4+** certificate, in 2019 and **PCI HSM V3.0** certificate in 2021. Procenne continues its R&D studies to add new members to the ProCrypt family, which is its locomotive product.

Maslak, İstanbul

Eski Büyükdere Cad. No:7 Giz 2000 Plaza, Kat:12 Maslak-Sarıyer, İstanbul/Türkiye

Telefon: +90 212 691 63 63 **e-Mail:** info@procenne.com

Avcılar, İstanbul

İstanbul Teknokenti, İÜ Avcılar Kampüsü, No: 224, Avcılar, İstanbul/Türkiye

Telefon: +90 212 691 61 67 **e-Mail:** info@procenne.com

Çankaya, Ankara

Ufuk Üniv. Cad. No: 3 Paragon Tower, Kat: 23 Kızılırmak Çankaya, Ankara/Türkiye

Telefon: +90 312 258 63 **e-Mail:** info@procenne.com

procenne procenneds

www.**procenne**.com