THALES
Building a future we can all trust

圣天诺 Envelope Plus

Windows应用与算法防护实战

李翔 | 高级解决方案顾问
泰雷兹圣天诺软件货币化

软件安全与保护的典型场景

过去的典型场景

未来场景

版权保护和反盗版

- 防止未经授权使用软件许可和产品功能
- 通过阻止逆向工程尝试、利用加密技术、反调试技术、混淆技术、完整性保护技术和其他安全技术来确保合规性。

知识产权保护

- 保护供应商的算法、逻辑和专有技术免受逆向工程和知识产权盗窃。

防止安全漏洞被检测和利用

- 通过增加漏洞的发现和利用难度，帮助供应商履行其**《网络弹性法案》(CRA)** 合规义务。
- 随着人工智能模型和工具（例如 Anthropic Mythos）的出现，漏洞检测和攻击手段的创建速度大大加快，这正成为一个非常紧迫的问题。

圣天诺 LDK Envelope Plus 有效地解决了所有这三个问题

圣天诺 在应对 Mythos 挑战中的作用

Mythos的威胁	软件开发商需要做的	圣天诺如何帮助您
Anthropic 的新型 Mythos 模型功能非常强大，可以轻松、快速且低成本地发现许多新的（通常非常复杂的）漏洞。	扫描、优先级排序和补丁速度更快	1. 源代码中的漏洞更容易被发现。 2. 软件开发商可利用新模型扫描其源代码，确定优先级并进行修补。 3. 我们大大增加了对已部署二进制文件（黑客可以访问的部分）进行逆向工程的难度。

关于Mythos的一些新闻：

苹果硬件安全防线被破解！Mythos+人类专家5天攻破最强硬件

Anthropic将就Mythos发现的网络安全漏洞向全球金融监管机构作通报

圣天诺 在应对 Mythos 挑战中的作用

> 测试结果： AI vs 圣天诺 Envelope Plus

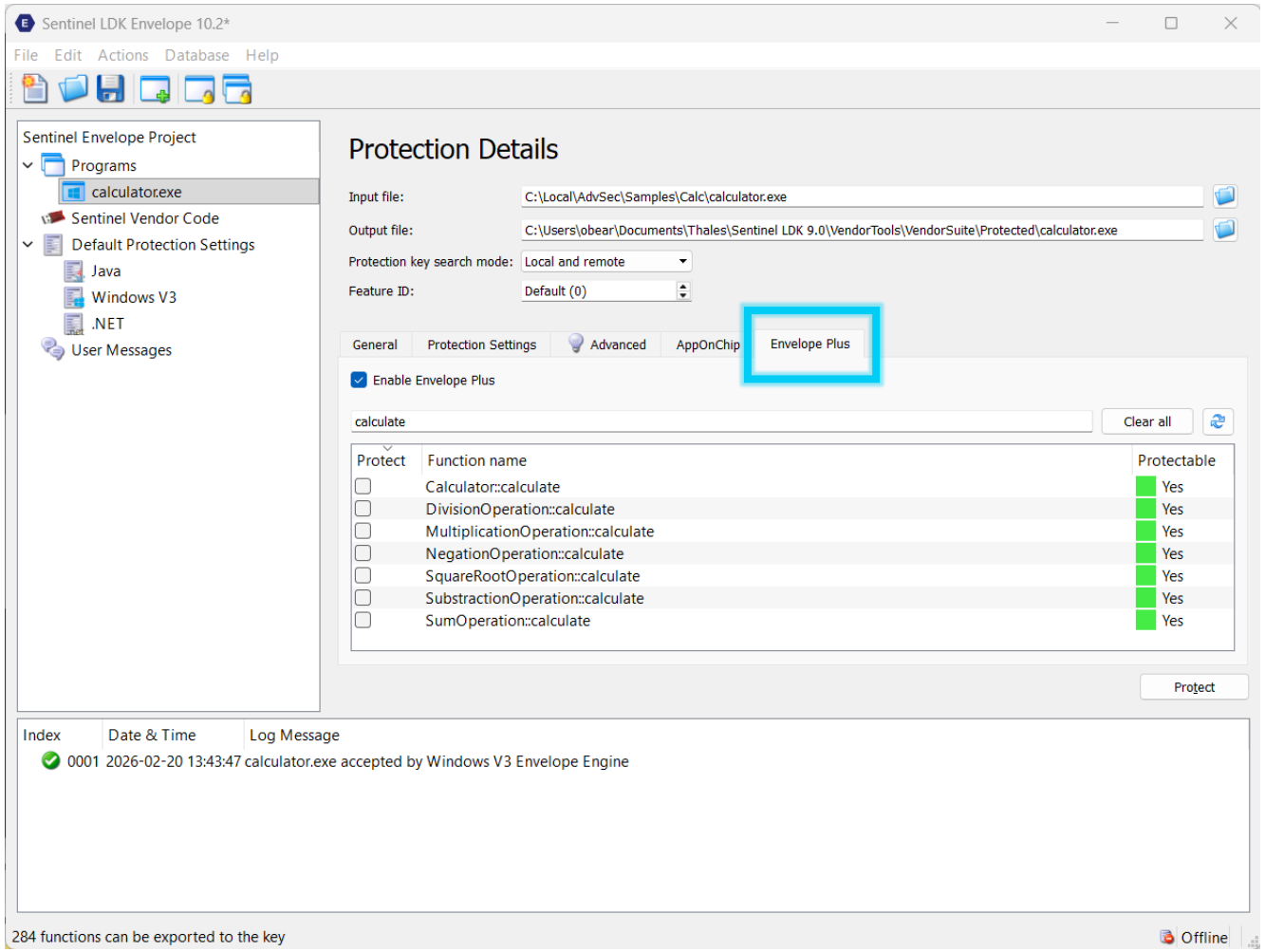
- Thales使用一个包含 10 个故意植入漏洞的 161KB 自定义字节码虚拟机进行了对比测试。通过使用Claude Opus 4.7 作为自主逆向工程代理，在两次测试中都使用了完全相同的初始条件：相同的Prompt、相同的初始工具，以及相同的外部工具使用权限。
- 唯一的变量是二进制文件是否受到圣天诺 Envelope 的保护。
- 针对未受保护的二进制文件，人工智能在 3 分 10 秒内，利用 342,000 个Token，在 38 行 log 中发现了 **10 个漏洞中的 8 个。**
- 针对受 Envelope 保护的二进制文件，人工智能请求了三个额外的工具，自行构建了一个自定义工具，生成了 2669 行log，消耗了 3.32 亿个Token，但**未发现任何漏洞。**经过六个多小时的分析，Opus 4.7 未发现任何漏洞，并**建议停止分析。**
- 当软件受到圣天诺 Envelope 保护时，人工智能难以触及应用程序逻辑并定位漏洞。缺少对软件逻辑结构的理解，自动化分析就无法进行。攻击者被迫退回到缓慢的手动攻击方式，从而抵消了人工智能辅助攻击所依赖的速度优势。
- 对于软件开发商而言，实际意义显而易见：保护措施可以争取时间进行检测、打补丁和部署修复程序，避免核心知识产权遭到窃取，从而避免损失。

Windows Envelope的高级IP保护

> **扩展圣天诺 LDK Envelope (32 和 64 位) 的现有功能，提供高级保护能力**

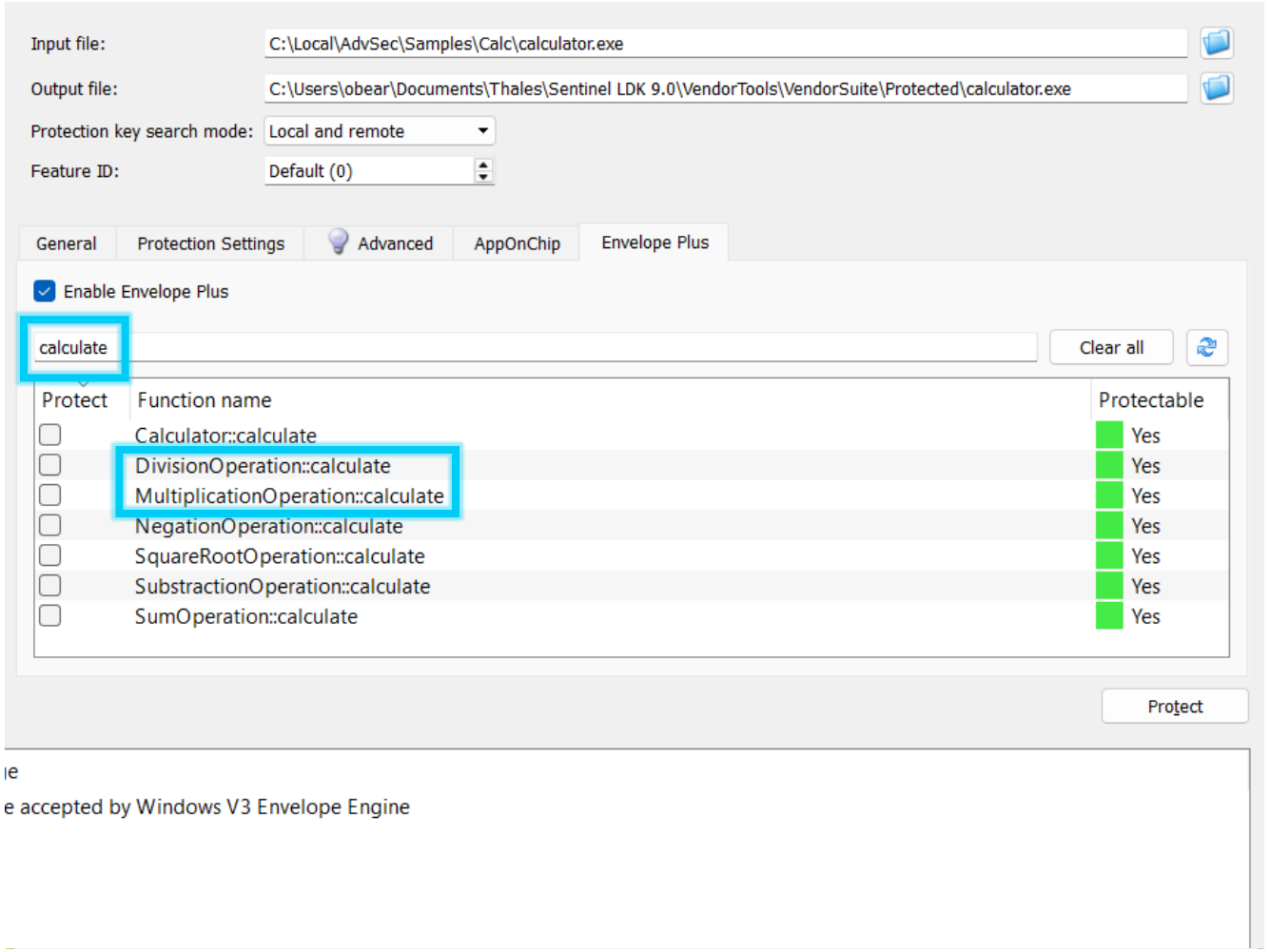
- 代码混淆
- 防篡改保护
- 防跟踪保护

> **满足绝大多数IP保护和反盗版应用需求**



Windows Envelope的高级IP保护

- **扩展圣天诺 LDK Envelope (32 和 64 位)的现有功能，提供高级保护能力**
  - 代码混淆
  - 防篡改保护
  - 防跟踪保护
- **满足绝大多数IP保护和反盗版应用需求**
- **易于使用的基于图形用户界面的保护功能选择**
- **实现了创新的代码提取和重新编译技术**
- **无需开发商提供源码,将DLL及EXE中重要函数提取并进行二进制转换**



圣天诺 LDK Envelope Plus的功能增强(C/C++)

	圣天诺 LDK Envelope	圣天诺 LDK Envelope w/Plus
数据文件加密	✅	✅
反debug调试	✅	✅
完整性检查	✅	✅
反非法拷贝	✅	✅
防止IP盗窃	✅	✅ (增强)
代码混淆	❌	✅
防篡改保护	❌	✅
防跟踪保护	❌	✅

代码混淆

> 将选定受保护函数的代码流转换为等效代码，

该等效代码产生相同的结果，但更难理解和遵循。

原始代码

```
double Solve(double a, double b, double c, double x)
{
    return a * x * x + b * x + c;
}
```

混淆

混淆后的代码*

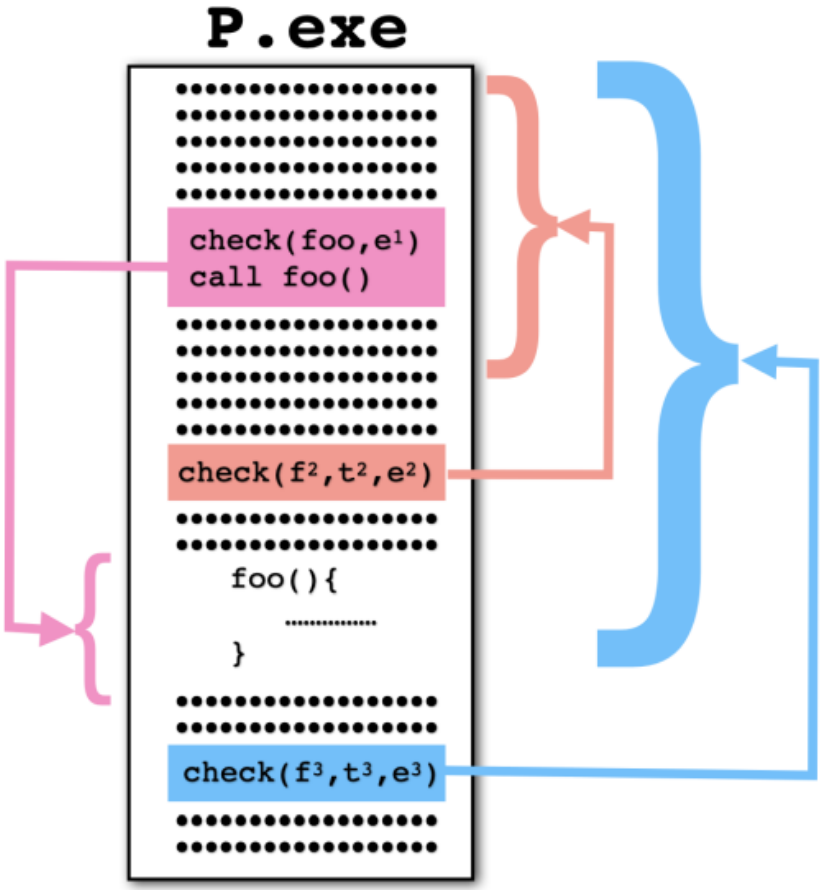
```
double F(double a, double b, double c, double x)
{
    int state = 0;
    double result = 0;
    double temp = 0;

    while (true)
    {
        switch (state)
        {
            case 0:
                temp = b * x + c;
                state = 1;
                break;
            case 1:
                result = a * x * x + temp;
                state = 2;
                break;
            case 2:
                return result;
        }
    }
}
```

* 注：这只是一个示例，为了清晰起见进行了简化。它并不代表 Envelope Plus 的实际混淆效果。

防篡改保护

- 防篡改保护是一种高级的完整性检查形式
- 完整性检查器（通常有数百个）会在程序启动时以及可执行文件中的多个随机位置插入，它们会计算随机（重叠）代码范围的哈希值并进行检查。
  - 完整性检查器本身受到混淆和反追踪技术的保护。
- 检查器会检测可执行文件是否已被修补以及正在运行的程序在内存中是否已被更改。



防追踪保护

> 防止攻击者执行动态程序分析

> 防止通过观察、记录和逐步执行程序，来理解或修改其行为。

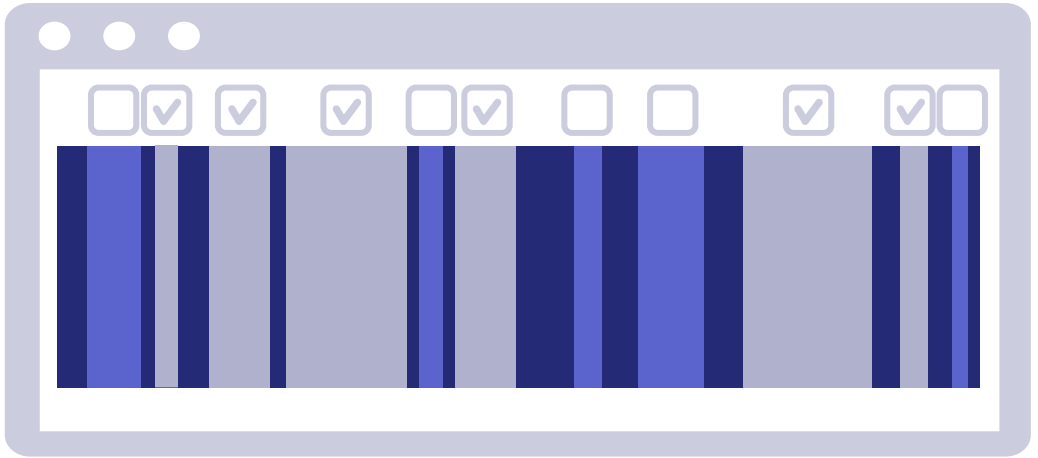
	Envelope 反 debugging	Envelope Plus 反跟踪
阻止标准debug调式器	✅	✅
防止受到常用二进制插桩框架(binary instrumentation frameworks)的攻击	❌	✅
已集成到 Envelope 运行时代码中，该代码在应用程序启动时执行。	✅	✅
集成到供应商自身的代码中，该代码在整个运行时执行。	❌	✅

Envelope Plus 如何工作

1. Envelope 识别到函数中的代码

Envelope Plus 如何工作

1. Envelope 识别到函数中的代码
2. 用户选择需要进行保护的函数



Envelope Plus 如何工作

1. Envelope 识别到函数中的代码
2. 用户选择需要进行保护的函数
3. 将选中函数代码提取到 LLVM-IR

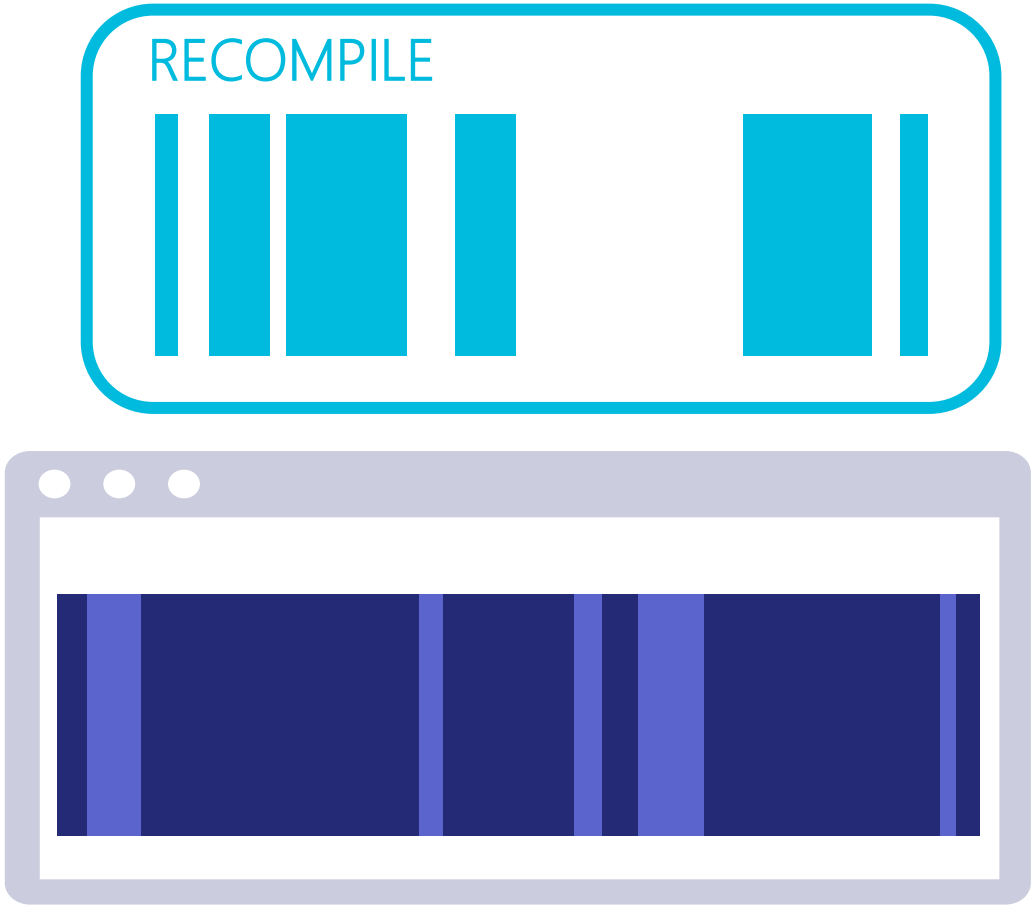
Envelope Plus 如何工作

1. Envelope 识别到函数中的代码
2. 用户选择需要进行保护的函数
3. 将选中函数代码提取到 LLVM-IR
4. Envelope Plus 对选中代码进行保护:

- 代码混淆
- 插入防篡改保护代码
- 插入防跟踪代码

Envelope Plus 如何工作

1. Envelope 识别到函数中的代码
2. 用户选择需要进行保护的函数
3. 将选中函数代码提取到 LLVM-IR
4. Envelope Plus 对选中代码进行保护:
   - 代码混淆
   - 插入防篡改保护代码
   - 插入防跟踪代码
5. 代码重编译

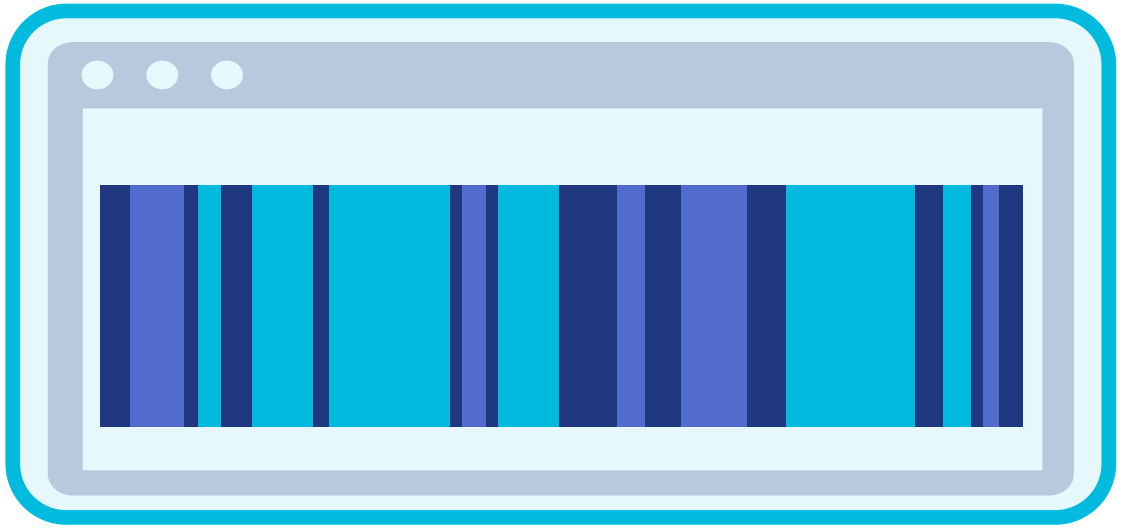


Envelope Plus 如何工作

1. Envelope 识别到函数中的代码
2. 用户选择需要进行保护的函数
3. 将选中函数代码提取到 LLVM-IR
4. Envelope Plus 对选中代码进行保护:
   - 代码混淆
   - 插入防篡改保护代码
   - 插入防跟踪代码
5. 代码重编译
6. 将代码重新插入原应用中

Envelope Plus 如何工作

1. Envelope 识别到函数中的代码
2. 用户选择需要进行保护的函数
3. 将选中函数代码提取到 LLVM-IR
4. Envelope Plus 对选中代码进行保护:
   - 代码混淆
   - 插入防篡改保护代码
   - 插入防跟踪代码
5. 代码重编译
6. 将代码重新插入原应用中
7. 应用圣天诺 LDK Envelope 其他保护功能:
   - 数据加密
   - 反debug调试
   - 数据文件保护
   - License 检查 植入



圣天诺 Envelope Plus的主要特性

文件加密	对二进制可执行文件进行加密，可以防止静态分析工具（如反汇编程序（针对本地代码）和反编译程序（针对托管代码））;对文件进行静态保护。
反debug调试	阻止调试器（黑客常用的工具）附加到进程，使动态分析（运行时）的使用更加困难。
完整性检查	防止可执行文件和二进制文件被试图绕过许可检查的黑客篡改。此机制通过一次检查评估整个文件是否存在更改。
专为防非法拷贝而设计	旨在通过在受保护的应用程序和License之间建立牢固的绑定关系来防止软件盗版。如果软件无法被修改以移除对许可证的依赖，即使攻击者能够查看其逻辑并了解软件的工作原理，也能成功实现防盗版。
专为知识产权保护而设计	旨在防止攻击者理解知识产权（应用程序逻辑和算法），从而阻止他们修改应用程序或将知识产权集成到自己的软件中。如果软件逻辑无法被理解或修改，无论是否强制执行许可协议，都能成功实现知识产权保护。即使在不强制执行许可协议且供应商不担心盗版的情况下，这种方法也很有用。此外，它还可以通过增加许可检查的逆向工程难度来加强复制保护。
反篡改保护	与完整性检查类似，防篡改可以检测二进制文件的更改，但它要复杂得多，由数百个不同的检查器执行，这些检查器会检查二进制文件的不同（有时甚至是重叠的）部分。
反追踪	与反调试类似，反追踪可以防止动态分析。它会插入代码，使追踪已执行的指令变得更加困难。这种保护不仅在程序启动时插入，还会插入到软件开发商自身的函数中。它能够抵御标准调试器之外的其他工具，例如二进制插桩框架。
代码混淆	这种软件保护技术将代码转换成复杂且混乱的格式，虽然能产生完全相同的结果，但却大大增加了理解难度。其主要目的是通过隐藏程序的原始逻辑和用途来抵御逆向工程。

THALES
Building a future we can all trust

圣天诺软件授权

更多案例与介绍
随时微信端咨询

www.thalesgroup-sm.cn