

基于动态二进制插桩的密钥安全性检测

林昊, 康绯, 光焱

(信息工程大学数学工程与先进计算国家重点实验室, 河南 郑州 450001)

摘要: 针对加密软件中的密钥安全性问题, 提出一种基于动态二进制插桩的密钥安全性检测方法。该方法面向 CryptoAPI 加密应用程序, 首先通过对 CryptoAPI 密钥应用模式的分析, 指出潜在的密钥安全性漏洞; 然后以动态二进制分析平台 Pin 为支撑, 动态记录程序运行期间的加解密过程信息; 在此基础上设计关联性漏洞检测算法, 实现对密钥安全性的检测。测试结果表明, 该方法能够有效检测出两大类密钥安全性漏洞。

关键词: 密钥安全性检测; 动态二进制插桩; Pin 平台; CryptoAPI 加密应用程序

中图分类号: TP309

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2017.00213

Key security detection based on dynamic binary instrumentation

LIN Hao, KANG Fei, GUANG Yan

(State Key Laboratory of Mathematical Engineering and Advanced Computing, PLA Information Engineering University, Zhengzhou 450001, China)

Abstract: For the key security problem in the cryptographic software, the method of key security detection based on dynamic binary instrumentation was proposed. Aimed at CryptoAPI cryptographic software, the method firstly pointed out the potential key security vulnerabilities by analyzing the key applying patterns of CryptoAPI. Then it recorded cryptographic data information during the execution of the program dynamically using Pin platform. On this basis, a relevance vulnerability detection algorithm was designed to detect the key security. Test result indicated that it can effectively detect the two kinds of key security vulnerabilities.

Key words: key security detection, dynamic binary instrumentation, Pin platform, CryptoAPI cryptographic software

1 引言

越来越多的应用程序采用加密手段来保护数据安全, 但是由于开发人员的编程失误等原因, 会导致加解密过程存在安全隐患。密钥是密码机制的核心环节, 然而近年来, 由于密钥的安全性所引发的各类漏洞和安全事件层出不穷, 因此, 对加密应用程序的密钥进行安全性检测在提高软件安全性、保护隐私数据等方面具有重要意义。

目前, 针对加密应用程序的密钥安全性检测主要通过两种方法。方法一是综合利用反汇编、二进制分析、软件动态调试等逆向分析手段^[1],

对加密应用程序的加解密过程分别进行静态语义分析和动态特征提取, 从中发现潜在的密钥安全性漏洞, 此方法的弊端是需要耗费大量人工分析的时间和精力。方法二是基于安全漏洞检测技术, 由于密钥安全性漏洞属于软件漏洞中密码学漏洞的范畴, 通过形式化证明、静态程序分析、动态软件测试^[2]等方法进行密码学安全性验证, 可以快速、有效地发现数据加解密过程中的密码学缺陷, 实现密钥安全性的检测。现阶段国内外的研究主要是利用方法二的密码学安全性验证的思想, 进行密码学应用安全性漏洞的自动化检测, 已有的检测工具主要有以下两种类型。

收稿日期: 2017-09-14; 修回日期: 2017-10-23。通信作者: 林昊, 284149270@qq.com

第一类工具是依靠静态程序分析和动态数据跟踪技术来实现安全性检测，代表性的检测工具有以下几种。2013 年，针对 Android 加密应用程序，Egele 等^[3]总结出 6 条密码学应用漏洞的规则，并实现了 CryptoLint 系统^[3]，该系统利用静态程序切片技术跟踪密钥、初始化向量等数据流向，并进行漏洞规则的检测，由于静态分析的约束，该系统会存在一定程度的误报。2014 年，Shao 等^[4]建立 Android 应用程序的密码学误用模型，并实现了动静结合的分析系统 CMA (crypto misuse analyzer)^[4]，该系统首先对 Android 应用程序做静态分析还原控制流图和密码函数调用图，以此指导应用程序的动态执行，并生成运行日志，在此基础上进行密码学安全性分析。李勇等^[5]则以静态程序扫描和动态跟踪分析为技术支撑。实现了面向 iOS 应用的密码学分析系统 iCryptoTracer，该系统仅针对密码学漏洞中的敏感数据泄露一类问题。除此之外，文献[6~9]也分别从静态程序分析和动态数据跟踪角度进行了分析研究。第二类工具则是以漏洞挖掘中的 Fuzzing 技术为支撑。例如，POET (padding oracle exploitation tool) 系统^[10]通过 Fuzzing 技术对 Web 应用的 Padding Oracle 攻击进行自动化检测；FIAT^[11]利用 Fault Injection 技术对密码设备进行安全性检测；2017 年的 Black Hat 会议上，Aumasson 等^[12]提出通过 Differential Fuzzing 技术来检测加密应用程序的安全性，并实现了自动化检测工具 CDF (crypto differential fuzzing)。

从上述研究现状中可知，虽然现阶段的密码学应用安全性漏洞检测已经取得了一定的研究成果，但其针对的目标问题相对单一、分散，没有专门针对密钥安全性的检测方案，且目前还没有完善的针对 Windows 平台应用程序的密码学应用安全性检测技术。

本文以 Windows 平台的 CryptoAPI 加密应用程序为研究对象，提出一种基于动态二进制插桩的密钥安全性检测方法。首先，对 CVE 信息库中的密钥安全性漏洞的成因机理进行分类分析；然后，分析 CryptoAPI 的 3 种密钥应用模式及其潜在安全性漏洞；最后，利用动态二进制插桩技术提取加解密过程信息，以此为基础设计关联性漏洞

检测算法，实现对密钥安全性检测。利用上述方法，检测出在通信软件 RTX 中存在着可预测的不安全密钥的漏洞，该漏洞会造成聊天信息的泄露。

2 密钥安全性漏洞分类

密码学漏洞是一类特殊的软件漏洞，通常会造成中间人攻击、旁路攻击和暴力破解攻击等^[13]危害，对软件安全、数据保密构成威胁，而由于密钥安全性导致的密码学漏洞更是威胁巨大，通常会造成严重的隐私数据泄露。本文对 NVD (national vulnerability database) 中 2011 年~2016 年期间 CVE 信息库中的密码学漏洞进行统计分析，结果如图 1 所示。

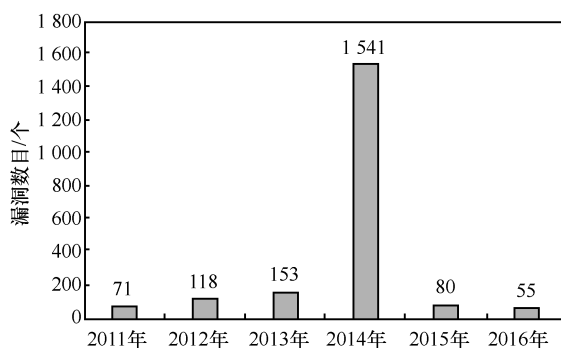


图 1 2011 年~2016 年 NVD 中密码学漏洞数量变化

为研究密钥安全性漏洞的成因，本文进一步从上述统计的密码学漏洞中提取出与密钥相关的 127 条漏洞记录，对造成密钥安全性漏洞的机理进行分析，并依据其机理进行分类与统计，结果如下所示。

1) 密钥自身为弱密钥而导致的安全漏洞

Key-Weak-Insecure 密钥的安全属性没有达到安全需求，包括密钥长度、密钥强度等。例如 CVE-2012-0655 漏洞，苹果操作系统 MacOS 的 libsecurity 库没有合理限定 X.509 证书中的 RSA 密钥长度，从而导致密钥存在安全性缺陷。

Key-Weak-Randomless 密钥未经随机数生成器处理，不具有足够的随机性，如直接取自字符串等方式都会导致此类漏洞。例如 CVE-2012-4366，Belkin 无线路由器基于 WAN MAC 地址的 8 bit 数字产生密码，导致此类漏洞。

Key-Weak-UnsafeRNG 密钥经过随机数生成器的处理，但随机数生成器自身存在安全缺陷。此

类漏洞有 CVE-2013-4442 和 CVE-2016-1618 等。

Key-Weak-Predicable 密钥可以被预测或产生密钥的随机数种子可预测, 包括无种子、固定种子、顺序/时间种子以及种子复用等情况。CVE-2011-0766、CVE-2012-2681 和 CVE-2013-5180 都属于此类漏洞。

2) 密钥应用过程不当而导致的安全漏洞

Key-ImproperUse-Reuse 不安全复用了密钥, 包括同一用户不安全复用了自身的密钥以及不同用户之间复用了相同的密钥等情况。例如在 CVE-2016-2364 漏洞中, Chrome 插件 HUDweb 在不同的安装用户之间使用了相同的私钥, 就属于此类漏洞。

Key-ImproperUse-Hardcoded 密钥被硬编码在程序中, 造成密钥可以被直接获取。CVE-2015-4080 漏洞中 AES 密钥和 CVE-2015-3960 漏洞中 RSA 私钥都存在此类漏洞。

Key-ImproperUse-Exposed 密钥的存储、传输等过程处理不当, 造成密钥敏感信息泄露。存储不当导致的安全漏洞如 CVE-2014-0199, 传输不当导致的安全漏洞如 CVE-2011-2223。

图 2 是不同类型的密钥安全性漏洞的数量统计。从整体上看, 应用过程的安全漏洞数量共 67 个, 由弱密钥导致的漏洞有 60 个, 前者要略高于后者。针对具体的漏洞类型, 密钥敏感信息泄露类型的漏洞所占比重最高, 由于密码机制的复杂性, 造成密钥敏感信息泄露的原因多种多样, 例如 CVE-2015-3322 漏洞是 Lenovo ThinkServer RD350

等系列服务器由于采用了弱算法来加密存储用户与管理员的 BIO 密码而导致密钥泄露; CVE-2011-2483 漏洞是由于未能正确处理 8 bit 特征从而导致密钥泄露。此类漏洞的成因复杂, 要实现对此类漏洞的检测相对困难。在统计结果中, 与随机数、随机数种子相关的 3 类漏洞 **Key-Weak-Randomless**、**Key-Weak-UnsafeRNG**、**Key-Weak-Predicable** 共占总漏洞数的 35%, 由此可见, 随机性缺陷是引发密钥安全漏洞的一个重要因素。

3 CryptoAPI 密码机制分析

在密码学的现实应用中, 应用程序通过调用各类密码算法库来实现隐私数据的保护、安全的数据通信。**CryptoAPI** 是目前应用最为广泛的密码库之一, 是微软针对 Win32 平台的应用程序, 为开发者提供了丰富的编程接口, 包括数据加解密、散列计算、数字签名和证书管理等功能, 这些功能函数为应用软件的开发提供了极大的便利。从勒索软件 **WannaCry**^[14]到通信软件 **FeiQ**, 都采用 **CryptoAPI** 来实现安全的数据加解密过程。

在 **CryptoAPI** 的机制中, 真正实现加解密操作的是独立的模块, 全称为加密服务提供者 (**CSP, cryptographic service provider**), 每个 **CSP** 有自己的密钥数据库, 统一管理加解密过程中的会话密钥、公私钥对。本文在对 **CryptoAPI** 密钥机制分析的基础上, 结合对大量样本中 **CryptoAPI** 调用信息的分析, 提取出在实际应用软件中, 密钥的 3 种主要应用模式, 图 3 为 3 种模式对应的密码函数流程。

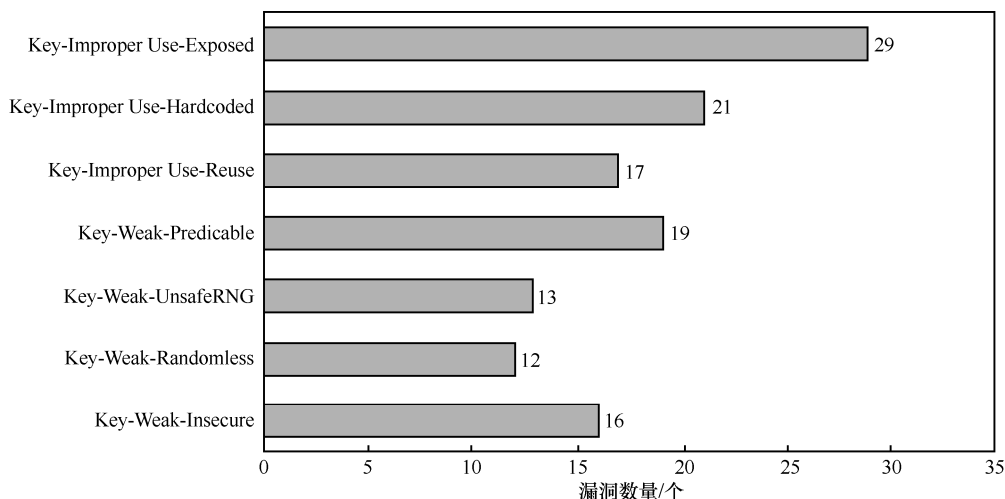


图2 密钥安全性漏洞机理的分类统计

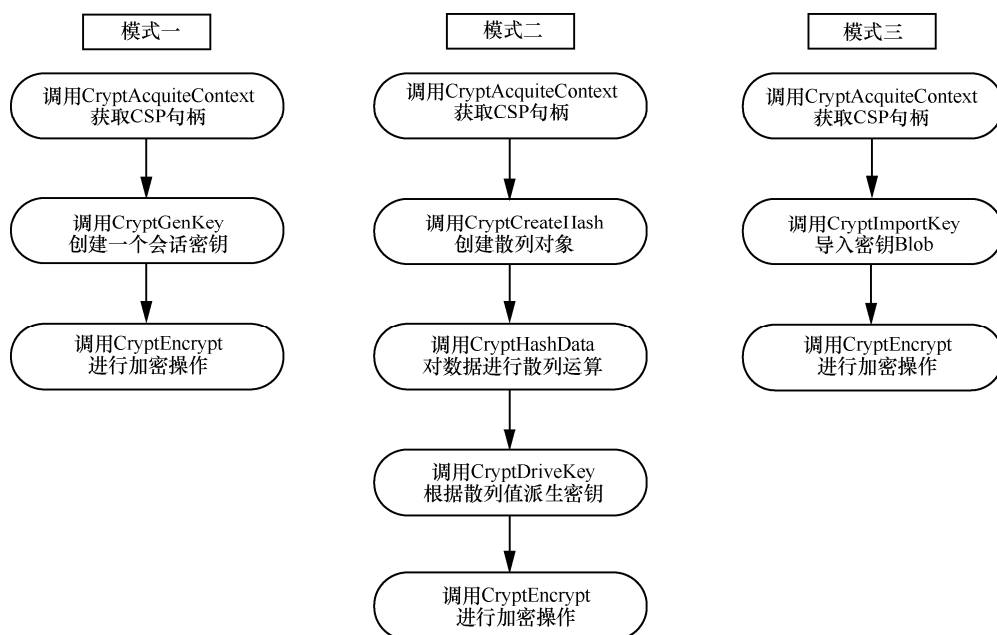


图 3 CryptoAPI 3 种密钥应用模式

模式一 通过调用接口函数在 CSP 内部产生

模式一的核心函数是 `CryptGenKey`，通过传入 CSP 句柄和相关标志位参数，在 CSP 内部生成指定算法的密钥，返回给用户密钥句柄。此密钥句柄是不透明的，即用户无法通过密钥句柄直接访问密钥的内容，只能利用此句柄进行加解密操作。密钥句柄的不透明特性，降低了密钥泄露的风险，在一定程度上提高了 CryptoAPI 体制的密钥安全性。

模式二 通过散列值派生出会话密钥

模式二将散列运算引入到密钥的生成过程中，其具体的调用模式如下：首先 `CryptCreateHash` 申请散列算法句柄，然后通过 `CryptHashData` 传入要进行散列计算的散列对象，最后通过 `CryptDeriveKey` 从散列运算结果中派生出指定算法的会话密钥，并返回密钥句柄。

模式三 通过接口函数从外部导入到 CSP 中

CryptoAPI 提供了 `CryptImportKey` 和 `CryptExportKey` 2 个接口函数完成密钥的导入导出操作。在对海量样本的分析中发现，应用程序会事先通过模式一或模式二生成密钥，然后通过 `CryptExportKey` 将密钥以 Blob 形式导出并保存。在软件真正执行过程中，直接通过 `CryptImportKey` 将事先存储的密钥 Blob 导入到 CSP 中，供应用程序使用。

3 种密钥应用模式分别存在不同类型的密钥

安全性漏洞。针对模式一，密钥在 `CryptGenKey` 函数内部生成，开发者无法自行指定密钥也无法干预密钥的内部生成过程，保证了密钥的安全性。但是，传入的参数 `dwFlags` 标志位必须满足对应密码算法的安全需求，否则会导致弱密钥的安全性漏洞，这种类型的漏洞在现实应用中的数量较少。对此类安全性漏洞进行检测，需要提取应用程序动态执行时 `CryptGenKey` 的 `dwFlags` 参数，然后对参数进行条件检查。在模式二中，进行散列计算的散列对象和散列算法将直接影响到密钥的安全性。与散列相关的密钥安全性漏洞出现较多，常见的漏洞有：每次散列计算的散列对象为固定值导致密钥可预测；散列计算的散列对象为数字和字母组合的简单字符串，且散列算法为 MD5 等已被证实是不安全的算法时，导致生成弱密钥。散列函数应用广泛，除了派生密钥外，还用于完整性检验、签名与验签等，而单一密码函数的参数检查无法实现对此类安全漏洞的检测，为此下文引入关联性检测，即对多个彼此存在关联关系的密码函数及其参数进行联合检查，从而实现漏洞检测。在模式三中最大的安全威胁是：一旦密钥 Blob 处理不当，就会造成密钥泄露。比如直接将会话密钥的 Blob 明文硬编码在应用程序中，或者 Blob 未经安全加密处理就进行本地存

储或网络发送,都会造成严重的密钥信息泄露,属于密钥应用过程不当而导致的漏洞。对于这类漏洞的检测,则需要借助动态污点分析技术^[15],跟踪密钥 Blob 的数据流,包括正向跟踪 CryptExportKey 导出密钥 Blob 的流向和逆向追踪 CryptImportKey 导入密钥 Blob 的来源。

基于以上分析, CryptoAPI 应用程序主要存在 Key-Weak-Insecure 和 Key-Weak-Predicable 两类弱密钥漏洞,本文接下来将围绕这两类密钥安全性漏洞进行检测。检测的工作包含 2 个环节: 1) 设计准确、高效的密钥安全性检测规则和检测算法; 2) 通过本节分析形成密码函数知识库,指导加解密过程信息的提取。

4 密钥安全性检测

4.1 检测规则的形式化描述

密钥安全性检测需要对加解密过程信息进行规则检测,本节设计了能够对密码函数、参数等信息进行规则检测的形式化描述方法。

定义 1 加解密过程

加解密过程 S 表示为软件动态执行的密码函数调用序列 $f_1, f_2, \dots, f_n$, 其中 f_n 表示为在第 n 个时刻的密码函数调用信息。每个 f_n 定义为三元组 $\{Addr, ApiName, Param\}$, 其中 $Addr$ 表示密码函数调用点地址, $ApiName$ 表示调用的密码函数名称, $Param$ 为参数集合 $\{P_i | i \in N\}$, 表示密码函数调用的参数信息。

定义 2 安全性漏洞检测规则

面向加解密过程的安全性漏洞检测规则 R 分为原子规则和复合规则。原子规则定义为三元组 $\{ApiName, Trigger, Relevance\}$, 其中 $ApiName$ 表示密码函数名称, $Trigger$ 表示触发因子, $Relevance$ 表示关联因子。复合规则定义为原子规则的组合。

定义 3 触发因子

触发因子 $Trigger$ 定义为集合 $\{<P_b, C_i> | i \in N\}$, 其中 P_i 表示所要检测的参数, C_i 表示安全性漏洞触发条件。触发因子表示对密码函数的特定参数进行相应的条件检查, 如果符合条件则表示激活触发因子, 满足了此条安全性漏洞检测的原子规则。

定义 4 关联因子

对于密码函数的分析发现, 密码函数的调用呈现关联性特征, 即为实现一个复杂的密码学功能, 需要由多个彼此关联的密码函数协同完成, 而这多个密码函数间通过特定的数据对象传递信息。图 4 所示是一次 CryptoAPI 的加密操作, 这是一种以句柄和句柄指针为媒介彼此关联、传递信息的方式。另外一种关联关系则是不同密码算法在参数取值上存在相等关系。因此, 有的安全性漏洞需要多个彼此关联的原子规则共同作用才会触发, 对于此类安全性漏洞的检测, 需要进行复合规则的检查, 而复合规则内的原子规则通过特定的关联因子彼此连接。

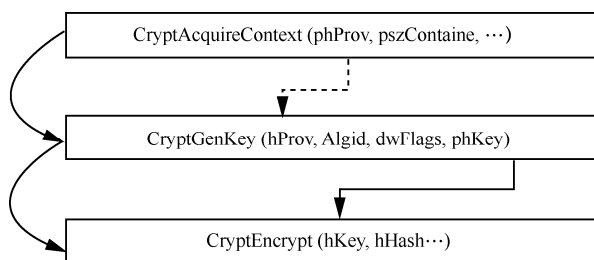


图 4 CryptoAPI 加密示例

关联因子 $Relevance$ 定义为三元组 $\{SFactor, DApi, DFactor\}$, 其中 $SFactor$ 表示源关联对象, $DApi$ 表示与之相关联规则的目标密码函数名称, $DFactor$ 表示目标密码函数的目标关联对象。

通过上述定义, 实现对安全性漏洞检测规则的形式化描述, 且该检测规则具有良好的通用性和扩展性, 不局限于对 CryptoAPI 的密钥安全性漏洞检测。将形式化的检测规则映射到 XML 规则描述文件中, 分析程序参照规则文件对加解密过程信息进行规则检测, 下面给出针对每条规则的安全性漏洞检测算法。

算法 1 安全性漏洞检测算法

输入: 动态提取的加解密过程信息 $f_1, f_2, \dots, f_n$, 检测规则 R

输出: 漏洞类型及漏洞点的密码函数、参数信息

算法开始

1) 依次从加解密过程信息中读取一条密码函数调用信息 f_i , 根据检测规则 R 进行检查。如果 R 是原子规则, 则执行 2); 如果 R 是复合规则, 则执行 3)。

2) 首先, 判断 *ApiName* 是否匹配, 检查触发因子 *Trigger* 是否匹配。若两者均匹配, 说明满足此安全性缺陷检测的原子规则, 输出漏洞信息, 然后转到 1) 进入下一次检查。如果不匹配, 则直接转到 1) 进行下一次检查。

3) 从复合规则中依次取出一条原子规则, 进行原子规则的检查。如果匹配, 则执行 4); 否则转到 1) 进入下一次检查。

4) 检查此条原子规则的关联因子 *Relevance*。如果 *Relevance* 非空, 则根据关联因子从加解密过程信息中搜索与之匹配的关联信息 f_j , 如果找到关联信息 f_j , 则以 f_j 为输入转到 3) 进行复合规则的下一条原子规则检查; 如果未找到关联记录, 则表明复合规则检查不通过, 转到 1) 进入下一次检查。如果 *Relevance* 为空, 则表明已完成复合规则的检查, 输出漏洞信息, 转到 1) 进入下一次检查。

算法结束

4.2 基于规则的密钥安全性检测

下面针对要检测的 *Key-Weak-Insecure* 和 *Key-Weak-Predicable* 两类密钥安全性漏洞, 用定义的形式化语言描述其检测规则。

$$R_4 = \begin{cases} \{CryptCreateHash, \{< AlgId, C_{Insecure} >, \{phHash, CryptHashData, hHash\}\} \\ \{CryptHashData, \{< pbData, C_{Insecure} >, \{hHash, CryptDeriveKey, hBaseData\}\} \\ \{CryptDeriveKey, NULL, NULL\} \end{cases} \quad (4)$$

漏洞 2 Key-Weak-Predicable

在散列值派生密钥的过程中, 如果密钥种子参数为固定种子, 则认为其派生的密钥是可预测的不安全密钥。此漏洞检测规则如下所示。

$$R_1 = \begin{cases} \{CryptHashData, \{< pbData, C_{Predicable} >, \{hHash, CryptDeriveKey, hBaseData\}\} \\ \{CryptDeriveKey, NULL, NULL\} \end{cases} \quad (5)$$

5 加解密过程信息的动态提取

加解密过程信息的提取是密钥安全性检测的基础, 本文通过动态二进制插桩技术提取应用程序动态执行时的加解密过程信息, 最终实现动态信息提取与离线规则检测相结合的密钥安全性检测方法。图 5 所示为动态信息提取的框架, 利用动态二进制平台, 监控应用程序调用密码算法库的过程, 按照函数参数库提取动态执行过程中的 API 名称、返回值和输入输出参数等信息, 并最终形成完整的加解密过程信息。

漏洞 1 Key-Weak-Insecure

漏洞 1 在 *CryptoAPI* 的应用中有 3 种具体表现形式。1) 在调用 *CryptGenKey* 函数生成指定算法密钥时, *dwFlags* 参数的高位 16 bit 指明密钥的长度信息, 如果不能正确通过参数指定长度信息, 便会导致生成弱密钥。2) 针对于密钥导入和导出函数而言, *pbData* 参数指向密钥 Blob 数据, Blob 具有特定的格式规范, 其包含有算法标志、密钥长度、密钥数据等数据成员, 因此需要检测 Blob 是否为弱密钥。3) 在以散列值派生密钥的模式中, 当散列运算的对象为数字和字母组合的简单字符串且散列算法为 MD5 等已被证实是不安全的算法时, 其派生的密钥则被认为是不安全的弱密钥。漏洞 1 对应的检测规则如下所示, 其中 R_1 、 R_2 、 R_3 为原子规则, R_4 为复合规则。

$$R_1 = \{CryptGenKey, \{< dwFlags, C_{Insecure} >, NULL\} \} \quad (1)$$

$$R_2 = \{CryptImportKey, \{< pbData, C_{Insecure} >, NULL\} \} \quad (2)$$

$$R_3 = \{CryptExportKey, \{< pbData, C_{Insecure} >, NULL\} \} \quad (3)$$

动态二进制插桩 (DBI, dynamic binary instrumentation) 技术^[16]指在不影响程序动态执行结果的前提下, 按照用户的分析需求, 在程序执行过程中插入特定分析代码, 实现对程序动态执行过程的监控与分析。目前, 应用广泛的动态二进制分析平台有 Pin^[17]、DynamoRIO^[18]和 Valgrind^[19]等。其中 Intel 公司的 Pin 平台提供了丰富的编程接口, 开发者可以通过调用编程接口方便地获取程序动态执行期间的指令、内存和寄存器等信息, 实现细粒度的动态监控, 因此本文选取 Pin 平台进行动态分析。

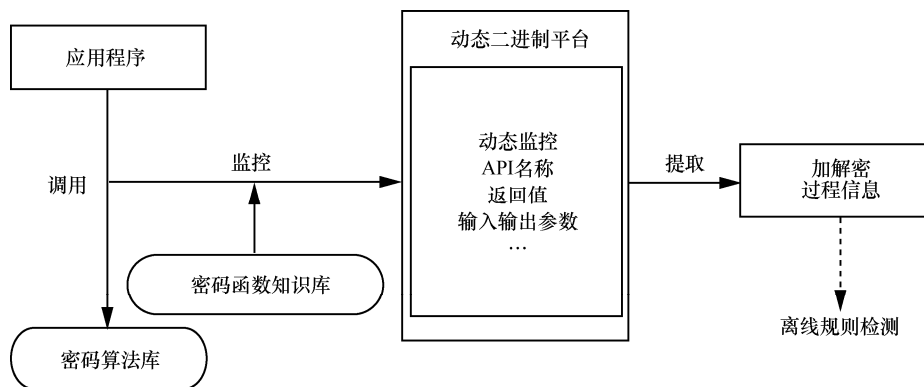


图5 动态监控记录的框架

Pin 提供了指令、基本块和系统调用等多个层次插桩分析，其中，RTN 是函数级的插桩机制，能够自动识别 API 函数；Trace 是轨迹层次的插桩机制，能够自动识别单一入口、多出口的指令轨迹块。本文通过 RTN 插桩和 Trace 插桩的结合，实现对 CryptoAPI 动态调用信息的获取。

算法2 加解密过程信息动态提取算法

输入 加密应用程序

输出 加解密过程信息 $f_1, f_2, \dots, f_n$

RTN 插桩模块

1) 监控每一次 API 调用，判断是否满足两个条件：是否需要分析的密码函数，是否是来自用户空间的函数调用。若满足条件则执行 2)，否则不处理。

2) 更新 *ApiRetAddr* 为当前密码函数调用的返回地址。

3) 依据函数参数库，获取当前动态执行状态的密码函数输入参数信息。

Trace 插桩模块

1) 拦截 Trace 轨迹，判断当前指令的地址空间，如果位于用户空间，则执行步骤 2)；否则不处理。

2) 获取 Trace 轨迹的入口指令的地址，与当前状态的 *ApiRetAddr* 匹配。若相等，则说明识别出当前动态执行的状态是对应密码函数的返回，此时清空 *ApiRetAddr* 并转到步骤 3)；否则不处理。

3) 依据函数参数库，获取当前动态执行状态的密码函数输出参数信息、返回值信息。

6 测试分析

6.1 功能测试

为验证基于动态二进制插桩的密钥安全性检测方法的正确性和有效性，本文针对 Key-Weak-Insecure 和 Key-Weak-Predicable 两类密钥安全性漏洞，选取 1 000 个包含 CryptoAPI 调用信息的样本进行测试分析，测试环境：Win7 x64 操作系统，Intel i7 处理器，16 GB 内存，2 T 硬盘，Pin 2.14-71313 动态二进制平台。

测试结果中，存在漏洞的样本比例达到 6%，其中 Key-Weak-Insecure 类漏洞有 5 个，Key-Weak-Predicable 类漏洞有 55 个，前者远少于后者，这也说明在实际 CryptoAPI 应用程序中更易发生由于密钥种子问题而导致密钥可预测的漏洞。对于存在密钥安全性漏洞的样本而言，其漏洞造成的实际危害取决于具体的加解密过程。下面以 RTX（腾讯通 2015）为例进行详细的分析说明。

RTX 是腾讯公司推出的企业级即时通信平台，提供消息发送、文本传输等多种办公功能。通过 RTX 提供的服务器和客户端软件，部署搭建测试环境，然后对客户端的登录、消息发送等过程进行检测分析，结果如表 1 所示。

表1 RTX 的检测结果

漏洞类型	加解密过程信息		
	调用点	函数名称	参数信息
Key-Weak-Insecure	sqlite3.dll 0x102B1	CryptHashData	RTX!
Key-Weak-Predicable	sqlite3.dll 0x102D2	CryptDeriveKey	Null

检测结果表明在 sqlite3.dll 的派生密钥过程

中，传入的密钥种子参数为固定的字符串“RTX!”，从而导致产生可预测的不安全密钥，且该密钥对于不同 RTX 用户而言是相同的。为了对检测结果进行验证，手动对 RTX 进行逆向分析，图 6 所示为密钥派生过程的反编译代码片段，代码显示通过对“RTX!”字符串进行 MD5 计算派生出 RC4 算法密钥，证实了检测结果的准确性。

```
12  if ( CryptCreateHash(hProv, 0x8004u, 0, 0, &u3) )
13  {
14      if ( CryptHashData(u3, "RTX!", 4u, 0) )
15          CryptDeriveKey(hProv, 0x6801u, u3, 0, &phKey);
16      CryptDestroyHash(u3);
17  }
18  result = phKey;
```

图 6 反编译代码片段

为了研究此漏洞会带来危害，进一步对加解密过程及通信机制进行逆向分析，跟踪不安全密钥的应用过程。通过分析发现，聊天信息的发送首先调用 Crypt.dll 的 oi_symmetry_encrypt2 函数进行对称加密，而该函数是 RTX 自实现并封装的 TEA 算法，聊天信息的发送过程并未用到上述不安全密钥。但是在信息发送之后，RTX 会利用不安全密钥对聊天信息进行 RC4 加密，并将密文存储到本地文件。

根据不安全密钥编写逆向解密过程，对本地的密文文件进行解密，图 7 所示为一段解密后的明文片段，包括聊天信息和字体信息等内容。由于不安全密钥在不同用户间是相同的，因此会导致本地加密存储的聊天信息可以被任何人解密，造成聊天信息泄露，是一种较为严重的隐私泄露。

```
41 <Content>
42 <Txt>你好，我是测试</Txt>
43 <Font Name="宋体" CharSet="134" PAF="0" Size="9" YOffset="0" Effects="1073741824" Color="0"/>
44 </Content>
```

图 7 密文文件解密片段

表 2 各种工具从技术手段和目标漏洞类型角度进行分析结果对比

工具	技术手段	目标漏洞类型
本文方法	动态二进制分析	密钥安全性漏洞
CryptoLint	静态程序切片	6 条漏洞经验规则
CMA	静态与动态混合分析	简单密码学误用
iCryptoTracer	静态扫描与动态跟踪	敏感数据泄露
POET	Fuzzing	Padding Oracle Attacks
FIAT	Fault Injection	Side Channel Attacks
CDF	Differential Fuzzing	逻辑漏洞

6.2 性能对比分析

将本文方法与现有的典型工具从技术手段和目标漏洞类型 2 个角度进行对比分析，结果如表 2 所示。

通过表 2 中的数据可以看出，相较于 CryptoLint 而言，本文所采用的动态分析方法准确更高；CMA 和 iCryptoTracer 具有动静结合的混合分析优势，但是 iCryptoTracer 仅针对单一的敏感数据泄露问题，CMA 检测的密码学误用相对简单，不能实现复杂问题的深度检测，而本文对密钥安全性的检测显然更加深入、全面；POET、FIAT 和 CDF 均采用了 Fuzzing 技术做支撑，但它们面向的目标漏洞类型仅针对特定类别的攻击或漏洞，而本文的方法能够覆盖整个密钥安全性漏洞的检测。另一方面，本文对检测规则的形式化描述方法通用性强，可以进一步应用于数据加解密过程的其他安全性检测。

7 结束语

本文首先对 CryptoAPI 密码机制进行分析，然后给出了密钥安全性检测规则的形式化描述，最后讨论了基于动态二进制插桩的密钥安全性检测方法。实验测试表明，本文的方法对于 CryptoAPI 应用程序具有很好的检测效果，能够正确地提取应用程序动态运行时的加解密过程信息，并且在此基础上能够准确地检测加解密过程中潜在的密钥安全性漏洞，通过与其他检测工具

之间进行对比,体现了本文方法的优越性。

由于动态分析的限制,当动态执行没有触发加解密函数时,就无法获取加解密过程信息进行安全性检测。提高动态分析的路径覆盖率,需要以样本的完整性分析技术为支撑,能够实现环境感知、敏感分支点识别判断,尤其在恶意代码分析领域,其采用的反检测、反调试等技术手段,使行为分析缺乏完整性,加解密行为无法被激发。针对样本的完整性分析是当前研究的一大热点、难点问题。

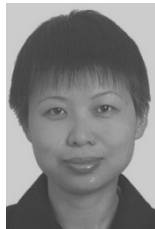
参考文献:

- [1] 段钢.加密与解密(第三版)[M].北京:电子工业出版社,2008.
DUAN G. Encryption and decryption(Third Edition)[M]. Beijing: Electronic Industry Press,2008.
- [2] 李舟军,张俊贤,廖湘科,等.软件安全漏洞检测技术[J].计算机学报,2015,38(4):717-732.
LI Z J,ZHANG J X,LIAO X K,et al.Software security vulnerability detection technology[J].Chinese Journal of Computers,2015,38(4):717-732.
- [3] EGELE M, BRUMLEY D, FRATANONIO Y, et al. An empirical study of cryptographic misuse in android applications[C]// ACM Sigsac Conference on Computer & Communications Security. 2013:73-84.
- [4] SHAO S, DONG G, GUO T, et al. Modelling analysis and auto-detection of cryptographic misuse in android applications[C]// IEEE International Conference on Dependable, Autonomic and Secure Computing. 2014:75-80.
- [5] LI Y, ZHANG Y, LI J, et al. iCryptoTracer: Dynamic Analysis on Misuse of Cryptography Functions in iOS Applications[C]// International Conference on Network and System Security. 2014:349-362.
- [6] MA S, LO D, LI T, et al. CDRep: automatic repair of cryptographic misuses in android applications[C]// ACM on Asia Conference on Computer and Communications Security.2016:711-722.
- [7] CHATZIKONSTANTINOU A, NTANTOGIAN C, KAROPOULOS G, et al. Evaluation of cryptography usage in android applications[C]// Eai International Conference on Bio-Inspired Information and Communications Technologies. 2016:83-90.
- [8] ALMEIDA J B, BARBOSA M, FILLIÀTRE J C, et al. CAOVerif: an open-source deductive verification platform for cryptographic software implementations[J]. Science of Computer Programming, 2014, 91(91):216-233.
- [9] FAHL S, HARBACH M, MUDERS T, et al. Why eve and mallory love Android: an analysis of android SSL (in)security[C]// ACM Conference on Computer and Communications Security. 2012:50-61.
- [10] RIZZO J, DUONG T. Practical padding oracle attacks[C]// Usenix Conference on Offensive Technologies. 2010:1-8.
- [11] BARENGHI A, BREVEGLIERI L, KOREN I, et al. Fault injection attacks on cryptographic devices: theory, practice, and counter-measures[C]// Proceedings of the IEEE.2012:3056-3076.
- [12] Automated testing of crypto software using differential fuzzing[EB/OL].<https://www.blackhat.com/docs/us-17/wednesday/us-17-Aumasson-Automated-Testing-Of-Crypto-Software-Using-Differential-Fuzzing.pdf>.
- [13] LAZAR D, CHEN H, WANG X, et al. Why does cryptographic software fail? A case study and open problems[C]// Asia-Pacific Workshop. 2014:1-7.
- [14] 安天针对勒索蠕虫“魔窟”(WannaCry)的深度分析报告[EB/OL].<http://www.antiy.com/response/wannacry.html>.
- [15] NEWSOME J, SONG D. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software[C]//Conference on NDSS. 2005.
- [16] NETHERCOTE N. Dynamic binary analysis and instrumentation or building tools is easy[D]. Trinity Lane, Cambridge: University of Cambridge. 2004.
- [17] LUK C K, COHN R, MUTH R, et al. Pin: building customized program analysis tools with dynamic instrumentation[J]. ACM Sigplan Notices, 2005, 40(6):190-200.
- [18] BRUENING D L. Efficient, transparent, and comprehensive runtime code manipulation[C]// Massachusetts Institute of Technology.2004.
- [19] NETHERCOTE N, SEWARD J. Valgrind:a framework for heavyweight dynamic binary instrumentation[C]// ACM Sigplan Conference on Programming Language Design & Implementation. 2007:89-100.

作者简介:



林昊(1992-),男,山东荣成人,信息工程大学硕士生,主要研究方向为网络信息安全。



康绯(1972-),女,河南周口人,信息工程大学教授,主要研究方向为网络信息安全。



光焱(1983-),男,安徽枞阳人,博士,信息工程大学讲师,主要研究方向为密码学、网络信息安全。