文章编号:2096-1472(2026)10-0067-06
DOI:10.19644/j.cnki.issn2096-1472.2026.010.011

面向内存取证的 DES 密钥搜索算法

黄景昊，吴礼发

（南京邮电大学计算机学院、软件学院、网络空间安全学院，江苏 南京 210023）
✉ 1843976722@qq.com; wulifa@njupt.edu.cn

摘 要:针对现有内存取证工具及研究多聚焦于现代加密算法的密钥提取,对传统 DES 密钥缺乏针对性搜索能力,导致处理 DES 加密数据时取证效率与完整性受限等问题,提出了一种面向内存取证的 DES 密钥搜索算法。首先,基于 OpenSSL 库中 DES 密钥的反混淆机制,解析其初次字节调换与二次位运算的混淆逻辑,通过逆向循环移位、字节置换及滑动窗口定位,将字节混淆序列还原为标准子密钥;其次,通过汉明距离结构验证方法,利用 DES 轮密钥循环群关系,通过 PC-2 逆置换、循环右移及汉明距离计算验证候选密钥合法性,结合位填充补全置换丢失位;最后,通过熵阈值过滤策略,基于 DES 子密钥高熵特性,选定熵阈值最优值。实验结果表明,该算法的搜索性能良好,在多次实验中误检率均为 0,漏检率在选取最优熵阈值的情况下仅有 7%。

关键词:密钥搜索;内存取证;DES 密钥
中图分类号:TP915.08 **文献标识码**:A

A DES Key Search Algorithm for Memory Forensics

HUANG Jinghao, WU Lifa

(*School of Computer Science, School of Software, School of Cyberspace, Nanjing University of Posts and Telecommunications, Nanjing* 210023, *China*)
✉ 1843976722@qq.com; wulifa@njupt.edu.cn

Abstract: Existing memory forensics tools and studies mainly focus on key extraction of modern encryption algorithms, yet they lack targeted search capability for traditional DES keys. This restricts the efficiency and completeness of forensic analysis when processing DES-encrypted data. To address this issue, a DES key search algorithm for memory forensics is proposed. First, based on the deobfuscation mechanism of DES keys in the OpenSSL library, the obfuscation logic of the initial byte swapping and secondary bitwise operations is analyzed. The byte-obfuscated sequence is restored to standard subkeys through reverse circular shifting, byte permutation, and sliding window positioning. Then, a Hamming distance structural verification method is introduced. By exploiting the cyclic group relationship of DES round keys, the validity of candidate keys is verified through inverse PC-2 permutation, circular right shifts, and Hamming distance calculation, combined with bit padding to recover the bits lost during permutation. Finally, an entropy threshold filtering strategy is employed, where the optimal entropy threshold is selected based on the high-entropy characteristic of DES subkeys. Experimental results demonstrate that the algorithm achieves excellent search performance: the false positive rate remains 0 across multiple experiments, and the false negative rate is only 7% when the optimal entropy threshold is applied.

Key words: key search; memory forensics; DES key

0 引言(Introduction)

内存取证软件的出现主要是为了解决传统数字取证方法在面对高级持续性威胁(Advanced Persistent Threat, APT)、无文件恶意软件、零日漏洞利用等复杂攻击时的局限性。在内存取证中,密钥搜索是一个关键方向,尤其是在处理加密数据和恶意软件时。调查人员使用内存取证从被黑客入侵的计算机中提取有用的数据和证据。威胁行为者使用各种策略来避免检测,而没有文件的恶意软件尤其难以检测。通过提取内存中的密钥,可以解密恶意软件的内部数据,了解它的行为模式、加密通信内容等。目前常见的内存取证工具如 Volatility[1]、WinPmem[2]等,虽然能够提取内存镜像并分析其中的部分信息,但大多只是对内存进行简单获取与分析,缺乏对加密密钥

收稿日期:2025-11-06

www.rjgczz.com

的针对性分析和搜索。近年来，研究者提出了多种面向内存取证的密钥搜索算法。Dolan-Gavitt 等[3]提出了一个基于PANDA的内存取证框架，针对操作系统及其应用程序的内存访问，挖掘浏览器活动中可能存在的密钥。Taubmann 等[4]提出了TLSkex来提取运行在虚拟机内部的进程的TLS密钥。他们利用虚拟机内省(Virtual Machine Introspection，VMI)技术来快照客户机内存，并使用限制性(限制为可写的内存页面和基于熵的查找)进一步减少搜索空间。DroidKex[5]提供了一种从应用于Android应用程序的进程内存中快速提取TLS密钥的方法。Halderman 等[6]提出了从内存中重建AES加密密钥的算法，该算法能够在几分之一秒内重建出7%比特衰减的密钥，在30 s内重建出15%比特衰减的密钥。Moriconi 等[7]提出了X-Ray-TLS，使用内存更改重建算法从进程内存中提取TLS密钥。现有内存取证相关研究较少关注DES密钥的搜索识别，然而在很多应用场景下，DES算法仍然被用作加密算法。OpenSSL是一个功能强大的加密库，广泛应用于各种软件系统中，提供了包括DES在内的多种加密算法的实现。由于它的易用性和广泛的社区支持，许多开发者在实现数据加解密功能时会选择使用OpenSSL库中的DES相关函数。现有的内存取证工具和方法可能无法直接适用于OpenSSL加密库中DES密钥的搜索。这导致在面对使用DES加密的数据时，传统的内存取证手段往往难以快速、准确地提取出DES密钥，从而影响了取证的效率和完整性。针对上述问题，本文设计了一种面向内存取证的DES密钥搜索方法，主要贡献如下：

(1)设计了一种基于OpenSSL加密库的DES密钥逆向方法，相较于传统的DES加密密钥，OpenSSL对DES加密进行了混淆，本文方法实现了逆混淆算法，并进行了还原，提升了识别效率。

(2)设计了一种基于汉明距离验证DES密钥结构的方法，通过计算当前候选轮密钥与上一轮候选轮密钥之间的汉明距离，快速判断当前候选轮密钥是否符合DES密钥子结构，提升了DES密钥识别准确度和效率。

(3)提出一种基于熵阈值的DES密钥过滤方法，通过设定特定的熵值阈值区分内存中可能的DES密钥数据与非密钥数据，有效提高识别的效率，降低了漏检率。

1 DES密钥特征分析(DES key feature analysis)

DES的输入密钥是64位长，但DES实际使用的密钥只有56位长。每个字节中的最低有效位是奇偶校验位，然后这些奇偶校验位被忽略，只使用每个字节的7个最高有效位，所以密钥长度为56位。该算法经过16次迭代，将明文块与从密钥获得的值交错，通过一系列步骤将64位输入转换为64位输出。

DES算法采用了每个子密钥生成的方式增加密钥的复杂性和安全性。每个轮子密钥都由主密钥(64位)通过密钥调度算法生成。密钥调度算法可以将64位的主密钥分成16个子密钥，每个子密钥48位，用于在每轮加密中与输入数据进行异或运算。

DES子密钥生成过程分为以下几个部分，DES子密钥生成的流程图如图1所示。

(1)将64位主密钥经过置换选择1(Permuted Choice 1简写为PC-1)后输出了56位，将它分为左右2个28位的数据块，分别记为C_0和D_0。PC-1置换函数是一个固定的置换表。

(2)对C_0和D_0进行循环左移操作。循环左移完成后生成C_1和D_1。因此，在16个轮次的计算当中会得到16个32位的数据块$C_1 \sim C_{16}$和$D_1 \sim D_{16}$。在DES中循环左移遵循着固定的规则：对于$i=1,2,\cdots,16$，对于C_i和D_i，若i为1、2、9或16，则循环左移一位，否则循环左移两位。

(3)对于C_1、D_1，将它们经过置换选择2(Permuted Choice 2简写为PC-2)后，得到48位的子密钥K1，用于每轮加密中与输入数据进行异或运算。PC-2置换的输入是由PC-1置换生成的56位的密钥，而输出是48位的子密钥。PC-2置换将56位的密钥重新排列，丢弃了8位并选取了其中的48位作为子密钥。

(4)至此，经过PC-2后的结果就是当前轮次的子密钥K_1了。在整个DES加密过程中会生成16个48位子密钥$K_1 \sim K_{16}$，分别用于DES算法中的16轮加密过程，从而保证每轮加密所使用的密钥都不同。

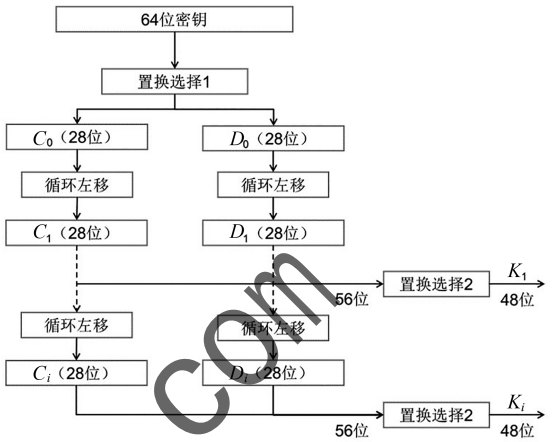

图1 DES子密钥生成的流程

Fig. 1 Process of DES subkey generation

2 面向内存取证的DES密钥搜索算法(DES key search algorithm for memory forensics)

2.1 算法思路

面向内存取证的DES密钥搜索算法架构如图2所示，该图展现了算法从内存数据获取到初始密钥还原的完整流程。该算法的核心思路是利用DES算法的特性，即在加密过程中会生成16轮子密钥用于加密操作，这些子密钥在加密过程中暂时存储在内存中。由于DES算法的子密钥生成过程是基于固定的规则，每一轮的子密钥都具有独特的结构特征，且这些子密钥之间存在明确的逻辑关系。

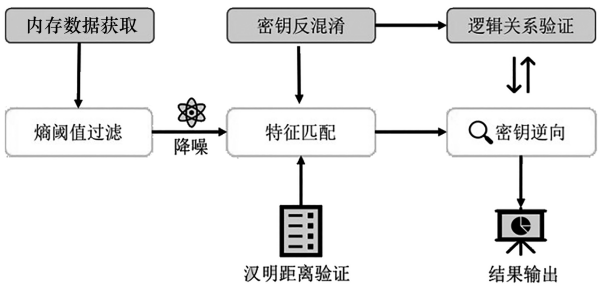

图2 算法架构

Fig. 2 Algorithm architecture

因此，通过内存取证的方法，获取待查内存，通过熵阈值过滤可排除掉随机内存段和重复内存段。依据相应的规则进行反混淆和特征匹配逆向还原内存段，验证是否符合DES子密钥结构特征。根据逻辑关系验证判断内存中是否存在DES密钥，并进行密钥逆向还原，获取初始DES密钥。

2.2 内存转储

内存取证的关键技术主要包括内存获取和内存分析两方面。内存获取是内存取证的第一步，也是最关键的一步，转储

www.rjgczz.com

内存的大小和采集时间的准确性都会影响内存分析的效率和结果。LiME[8-9]是用于 Linux 系统的内存获取工具,但 LiME 只能转储整个电脑内存,无法针对某个进程进行独立转储,导致转储文件过大严重影响内存分析的效率。Linux 中的 gcore 命令是一个用于生成运行中进程核心转储文件(core dump)的工具。该命令可以在不中断进程运行的情况下,生成一个包含进程当前内存状态的核心转储文件。

为了获取存在 DES 子密钥生成后的内存转储,需要严格控制内存转储的时机。Ottmann[10]提出了关于内存转储时机的 2 个一致性定义:瞬时一致性和准瞬时一致性。瞬时一致性描述了内存获取的理想情况,系统可以被冻结;准瞬时一致性是一个尊重因果顺序的概念,但与冻结系统没有区别。当内存转储是准瞬时一致性时,瞬时一致性可以在采集过程中找到一个时间点,在该时间点上存储器内容与采集的内容处于相同的状态。这意味着,尽管系统在转储过程中仍在运行,但采集到的内存转储文件仍然符合一致性的定义。基于上述定义,本文提出了一种转储方法:在 DES 子密钥生成后到加密结束前的这段时间内进行内存转储。这一时间段内,DES 子密钥完整地存在内存中,且尚未被覆盖或清除。通过在这一关键时间段内进行内存转储,可以保证采集到的内存转储文件中包含完整的 DES 子密钥结构,从而为后续的密钥提取和分析提供可靠的数据基础。这一方法不仅确保了内存转储的一致性,还提高了密钥提取的准确性和效率。

2.3 基于 OpenSSL 加密库的 DES 密钥逆混淆

2.3.1 初次混淆

研究中发现,标准 DES 算法生成的数据与 OpenSSL 加密库的 DES 算法生成的数据在内存中的存放顺序存在显著差异。通过对 OpenSSL 源码进行深入分析,发现 OpenSSL 加密库在生成 16 轮子密钥时进行了特殊的混淆处理。具体而言,OpenSSL 对部分字节码的顺序进行了重新排列,并通过一个名为 Schedule 的结构体来存放经过处理的子密钥,混淆后的内存数据会干扰子密钥结构特征验证,所以要对混淆的数据进行反混淆。通过对 Schedule 结构体的跟踪分析,标准 DES 算法字节码的排序和 OpenSSL 库 DES 算法混淆的字节码排序见表 1～表 4。DES 密钥为 8 个字节,t 为从原始密钥中取出前 4 个字节,s 为从原始密钥中取出后 4 个字节,经过 PC-1 变换和 PC-2 变换后为表中数据展示,从表中对比可知,OpenSSL 加密库将中间 2 个字节的顺序进行了调换,这是 OpenSSL 加密库的初步混淆。

表 1　标准 DES 加密库 PC-2 表置换后 t 字节码排序

Tab. 1 t bytecode sorting of PC-2 table after permutation in standard DES encryption library

字节序号	位序号	8	7	6	5	4	3	2	1
3	32～25			2	13	20	27	7	16
2	24～17			8	26	4	12	19	23
1	16～9			10	21	6	15	28	3
0	8～1			5	1	24	11	17	14

表 2　OpenSSL 加密库 PC-2 表置换后 t 字节码排序

Tab. 2 t bytecode sorting of PC-2 table after permutation in OpenSSL encryption library

字节序号	位序号	8	7	6	5	4	3	2	1
3	32～25			2	13	20	27	7	16
1	24～17			10	21	6	15	28	3
2	16～9			8	26	4	12	19	23
0	8～1			5	1	24	11	17	14

表 3　标准 DES 加密库 PC-2 表置换后 s 字节码排序

Tab. 3 s bytecode sorting of PC-2 table after permutation in standard DES encryption library

字节序号	位序号	8	7	6	5	4	3	2	1
7	32～25			32	29	36	50	42	46
6	24～17			53	34	56	39	49	44
5	16～9			48	33	45	51	40	30
4	8～1			55	47	37	31	52	41

表 4　OpenSSL 加密库 PC-2 表置换后 s 字节码排序

Tab. 4 s bytecode sorting of PC-2 table after permutation in OpenSSL encryption library

字节序号	位序号	8	7	6	5	4	3	2	1
7	32～25			32	29	36	50	42	46
5	24～17			48	33	45	51	40	30
6	16～9			53	34	56	39	49	44
4	8～1			55	47	37	31	52	41

2.3.2 二次混淆

在 OpenSSL 库的 DES 加密实现中,为了进一步增强加密算法的安全性,对子密钥生成过程进行了第二次复杂的混淆处理。这种混淆处理是在第一次混淆的基础上进行的,旨在通过增加额外的复杂性,使子密钥的生成过程更难以预测和破解。具体而言,OpenSSL 对初步混淆后的变量 s 和 t 进行了二次混淆操作。这一过程通过式(1)～式(4)实现,t 和 s 二次混淆后保存在子密钥缓冲区的格式见表 5 和表 6,表 5 和表 6 共同组成一个子密钥。

首先,将 t 左移 16 位,与 s 的低 16 位进行按位或操作,得到一个 32 位的中间变量 t_1,如式(1)所示:

$$t_1=((t\ll 16L)\mid(s\&0x0000ffffL)) \tag{1}$$

然后,使 t_1 循环右移 30 位,并将结果存储到子密钥缓冲区中,如式(2)所示:

$$*(k++)=\mathrm{ROTATE}(t_1\gg 30L)\&0xffffffffL \tag{2}$$

接下来,将 s 右移 16 位,与 t 的高 16 位进行按位或操作,得到另一个中间变量 s_1,如式(3)所示:

$$s_1=((s\gg 16L)\mid(t\&0xffff0000L)) \tag{3}$$

最后,将 s_1 循环右移 26 位,并将结果存储到子密钥缓冲区中,如式(4)所示:

$$*(k++)=\mathrm{ROTATE}(s_1\gg 26L)\&0xffffffffL \tag{4}$$

表 5　二次混淆后 t_1 字节码排序

Tab. 5 Bytecode sorting of t_1 after secondary obfuscation

字节名称	字节序号						
t	6	53	34	56	39	49	44
	4	55	47	37	31	52	41
s	2	8	26	4	12	19	24
	0	5	1	24	11	17	14

表 6　二次混淆后 s_1 字节码排序

Tab. 6 Bytecode sorting of s_1 after secondary obfuscation

字节名称	字节序号						
t	7	42	46	2	13	20	27
	5	40	30	10	21	6	15
s	3	7	16	32	29	36	50
	1	28	3	48	33	45	51

2.3.3 反混淆

在内存取证过程中,获取的数据是经过 OpenSSL 加密库二次混淆处理后的结果。为了还原出标准 DES 加密后的子密

www.rjqcz.com

钥，必须进行相应的反混淆操作。这一过程分为 2 个阶段：首先，对二次混淆进行逆向操作，恢复到初次混淆后的状态；随后，再对初次混淆进行逆向操作，最终得到标准 DES 加密后的子密钥。

内存取证获取到的内存快照由连续字节组成，其中每组 OpenSSL 加密库二次混淆后的字节序列长度为 128 字节。为了有效地进行反混淆操作，首先使用滑动窗口方法定位待查的 128 字节序列。随后，将该序列划分为 16 组，每组 8 字节。由于每组 8 字节是连续存放的，为了进行二次反混淆，需要将它进一步拆分为 2 个部分：t_1 和 s_1，分别对应不同的字节码。通过逆向循环移位和字节置换操作，逐步恢复初次混淆后的状态。这一过程通过式(5)～式(8)实现。

(1)对 s_1 进行循环左移 26 位操作，如式(5)所示：

$$s_1 = \text{REVERSE_ROTATE}(s_1 \ll 26L) \tag{5}$$

(2)对 t_1 进行循环左移 30 位操作，如式(6)所示：

$$t_1 = \text{REVERSE_ROTATE}(t_1 \ll 30L) \tag{6}$$

(3)将 s_1 右移 16 位，与 t_1 的高 16 位进行按位或操作，得到原始 s，如式(7)所示：

$$s = ((s_1 \gg 16L) \mid (t_1 \& 0xffff0000L)) \tag{7}$$

(4)将 t_1 左移 16 位，与 s_1 的低 16 位进行按位或操作，得到原始 t，如式(8)所示：

$$t = ((t_1 \ll 16L) \mid (s_1 \& 0x0000ffffL)) \tag{8}$$

本节针对 OpenSSL 加密库中 DES 密钥的初次混淆机制，提出了一种逆向方法，以恢复出标准的 DES 子密钥结构。OpenSSL 在生成子密钥时，采用了复杂的混淆规则(见表 1～表 4)，通过特定的位操作和置换操作对原始密钥进行混淆。为了逆向这一过程，根据这些混淆规则，对混淆后的字节序列进行精确的字节提取。依据表 1～表 4 中定义的混淆位位置，从混淆后的字节序列中提取出相应的位。这些位在混淆过程中被重新排列，因此需要按照逆向置换规则将它们还原到原始位置。置换完成后，将恢复后的 s 和 t 进行拼接，构成标准的 DES 子密钥。这一子密钥将用于后续的标准 DES 密钥结构验证，以判定其是否符合 DES 算法的规范。

2.4 基于汉明距离的 DES 密钥结构验证方法

2.4.1 方法原理

本文面向内存快照场景，提出了一种高效且准确的 DES 密钥提取方法，核心原理是通过分析内存中的数据，利用 DES 轮密钥之间的循环群(循环群指字节的循环位移)关系和低熵值特性，识别出可能的 DES 密钥。其核心思想是通过 PC-2 表逆向还原密钥，进行逆向循环后基于汉明距离对比轮密钥间的误差，判断数据是否符合 DES 轮密钥的特征，连续通过 16 轮验证即可判定为 DES 子密钥，再通过位填充和 PC-1 表还原出初始密钥。

汉明距离(Hamming Distance)是指 2 个等长字符串之间，在相同位置上不同字符的个数，也是将一个字符串转换为另一个字符串所需的最小替换次数。

汉明距离的数学公式如式(9)所示：

$$H(A,B) = \sum_{i=1}^{n} 1(a_i \neq b_i) \tag{9}$$

2.4.2 方法实现

DES 算法的原始密钥是一个 64 位的二进制串，去掉 8 位奇偶校验位，剩余 56 位参与运算。首先，通过 PC-1 表进行置换，转换成 56 位的密钥。接着，这 56 位的密钥通过循环左移操作，并再次应用 PC-2 表进行置换，最终生成 16 轮所需的 48 位子密钥。在这个过程中，初始密钥的一些位在变换中被舍弃。因此，从内存中搜索得到的密钥需要经过逆向还原恢复这些丢失位，才能还原成初始的 64 位密钥。逆向还原方法的设计为，通过滑动窗口，选取内存中的 16 轮二进制串，先通过熵滤波判断是否符合轮密钥的特征。若符合，取出第一轮和第二轮的二进制串进行 PC-2 表的逆向置换，将第二轮置换的结果循环右移和第一轮的结果进行异或，计算位之间的汉明距离。若汉明距离符合预期结果，则判断这两轮子密钥符合 DES 子密钥结构。因 DES 算法中相邻两轮子密钥存在循环群的关系，所以可以使用第二轮数据相应未丢失位填补第一轮数据中的丢失位。当 16 轮的轮密钥都通过预期检测时，将第一轮还原的结果经由 PC-1 表进行逆向还原。DES 初始密钥需要满足奇偶校验位的特性，即每个字节中位为 1 的个数需要是奇数个[8]。根据这个特性将逆向还原的结果进行二进制填充，最终还原出初始密钥并输出。以下以第一轮和第二轮的比较作示例，为简化说明，对 PC-2 表和密钥数据进行等比放缩。将 48 位的 PC-2 表和 56 位的还原数据等比放缩为 12 位和 14 位。等比放缩后的 PC-2 表，见表 7，基于汉明距离验证密钥结构的过程如图 3 所示。

表7 等比放缩的 PC-2 表

Tab. 7 PC-2 table with proportional scaling

2	10	7	5
3	1	11	13
8	6	14	9

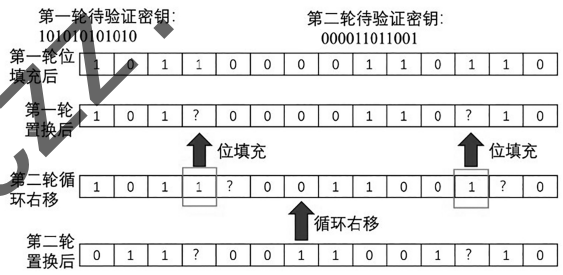

图 3 验证密钥结构示例

Fig. 3 Examples of key validation structure

从内存中取出第一轮待验证密钥和第二轮待验证密钥经 PC-2 表变换后形成图中的第一轮数据和第二轮数据，图中“?”的位置表示 PC-2 表逆还原后造成的丢失位，第二轮循环右移后与第一轮进行比较，通过汉明距离的检测进行位填充，补齐丢失位。

DES 子密钥的生成过程涉及两轮置换，每轮置换会导致 8 位信息的丢失。在此方法中，首先通过逆向操作补齐了第二轮置换中丢失的位，这一补齐后的数据已经通过了子密钥结构验证。所以，补齐后的数据即为经过第一轮置换(PC-1)后的标准 DES 密钥数据，所以对它执行 PC-1 的逆还原操作。在逆还原过程中，为了确保数据的完整性和准确性，对每个字节进行了位 1 检查。具体而言：如果某个字节中位 1 的个数为奇数，则将该字节的最后一位设置为 0；反之，如果位 1 的个数为偶数，则将该字节的最后一位设置为 1。这一步骤利用了 DES 密钥中奇偶校验位的特性，确保了还原后的密钥符合 DES 算法的规范。通过上述方法，成功地从内存快照中还原出了标准的 DES 密钥。

3 实验及结果分析(Experiments and result analysis)

为了验证算法的可行性，实验在具有 Intel(R) Core(TM) i7-9750H CPU @ 2.60 GHz 和 24 GB RAM 的电脑上进行。操作系统是 Ubuntu22.04。

www.rjgcz.com

3.1 获取内存转储

通过 OpenSSL 库使用 DES 加密的程序的内存转储过程简述如下:

(1)调用 OpenSSL 加密库编写程序,给定初始密钥,编译程序使之成为可执行文件。

(2)运行可执行文件并通过 Linux 命令"&"获取可执行程序的进程号。

(3)通过 gcore 命令抓取内存,生成内存转储文件(x-core)。

3.2 结果分析

3.2.1 熵阈值选取

熵(Entropy)是信息论中的一个基本概念,用于衡量信息的不确定性或随机性。对于一个离散随机变量 $\boldsymbol{X}$,它的熵 $H(\boldsymbol{X})$定义如式(10)所示:

$$H(X)=-\sum_{i=1}^{n}P(X_i)\mathrm{lb}P(X_i) \tag{10}$$

其中:$P(X_i)$是 $\boldsymbol{X}$ 取值为x_i的概率。

在内存取证中,即便通过进程号提取内存能够显著减小内存文件的大小,但在庞大的内存数据中定位加密密钥,就像大海捞针,因为 DES 子密钥结构只占用千兆字节数据中的 128 个字节。所以,本文提出了基于熵阈值过滤 DES 密钥的方法,通过滑动窗口选定 128 字节的数据,计算数据中重复字节的个数,当重复字节个数超过设定的阈值时,方法判定这段数据不符合 DES 子密钥的结构。这种方法的核心在于利用 DES 子密钥的高熵特性,通过简单的视觉测试即可区分出密钥区域。

本文通过编写脚本生成了 100 份和 500 份随机密钥,并利用 OpenSSL 库对这些密钥进行了加密处理,同时提取了加密过程中的内存使用情况。针对这两组密钥,分别在不同的熵阈值条件下进行了实验,以评估其性能表现。100 份随机密钥在不同熵阈值条件下的实验数据如图 4 所示,500 份随机密钥在相同条件下的实验数据如图 5 所示。

分析结果表明,随着熵阈值的增加,漏检率呈现出逐渐下降的趋势,这表明较高的熵阈值有助于提高密钥的检测准确性和安全性。然而,这种安全性的提升是以增加检测延迟为代价的。具体来说,当熵阈值从 5 增加到 20 时,漏检率从 40%(图 4)和 34%(图 5)降至 0,但检测延迟也从 32 ms 和 34 ms 增加到 118 ms 和 121 ms。

为了在安全性和效率之间找到一个平衡点,研究参考了潘昊斌等[11]关于熵检测阈值的研究。文献指出,选择适当的熵阈值可以降低误报率和漏报率,同时实现较小的检测延迟。基于这一理论,从图 4 和图 5 中可以看出:将熵阈值设定在 7 时,误检率均为 0,图 4 中的漏检率为 7%,检测延迟为 36 ms;图 5 中的漏检率为 7%,检测延迟也为 36 ms。这一设定能够在保证较低漏检率的同时,避免过长的检测延迟,从而在安全性和效率之间实现最佳平衡。

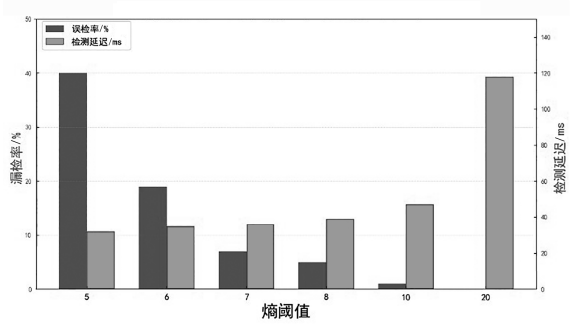

图 4　不同熵阈值条件下 100 份随机密钥的漏检率与运行时长对比

Fig. 4 Missed detection rate and detection delay of 100 random keys under different entropy threshold conditions

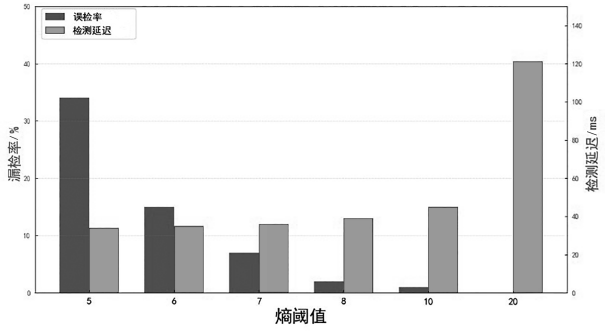

图 5　不同熵阈值条件下 500 份随机密钥的漏检率与运行时长对比

Fig. 5 Missed detection rate and detection delay of 500 random keys under different entropy threshold conditions

综上所述,本研究通过实验数据分析,提出了一个合理的熵阈值设定建议。这一建议不仅有助于提高密钥检测的准确性,还能有效控制系统的检测延迟。

3.2.2 误检率分析

确保检测工具能够在不包含密钥的随机内存文件中不产生误报至关重要,因为这关系到检测系统的可靠性和减少不必要的安全警报。本实验通过脚本生成了 500 份大小为 460 kB 的随机内存数据,并使用工具对这些内存数据进行检测。

随着熵阈值的增加,检测延迟呈现出显著的上升趋势,如图 6 所示,当熵阈值从 5 增加至 20 时,检测延迟从约 1000 ms 增加至超过 1200 ms。这一趋势表明,较高的熵阈值可能导致更长的检测时间,这是由于越高熵阈值所过滤掉的数据越少。但是,无论熵阈值如何变化,误检率始终保持在 0。这一结果表明,在本实验条件下,所采用的检测方法能够有效避免误检情况的发生,即使在熵阈值较高的情况下也不会产生误报,确保了检测算法的准确性和可靠性。

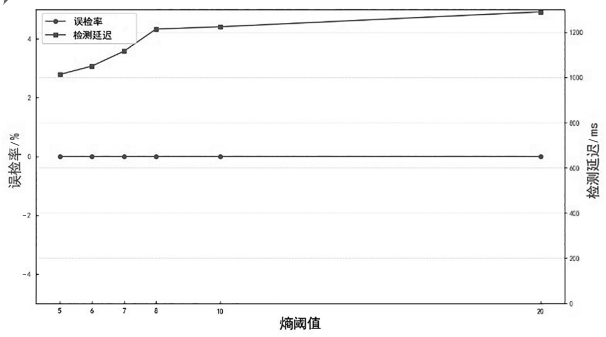

图 6　不同熵阈值下 500 份随机内存数据的误检率和检测延迟

Fig. 6 False detection rate and detection delay of 500 random memory data under different entropy thresholds

4 结论(Conclusion)

本文提出了一种面向内存取证的 DES 密钥搜索算法,旨在解决现有内存取证工具对传统 DES 密钥缺乏针对性搜索能力的问题。通过对 OpenSSL 库中 DES 密钥混淆机制的深入分析,设计了基于逆向循环移位、字节置换和滑动窗口定位的反混淆策略,成功将混淆后的字节序列还原为标准 DES 子密钥。结合汉明距离结构验证方法,利用 DES 轮密钥间的循环群关系和 PC-2 逆置换操作,实现了对候选密钥的高效验证,并通过位填充技术补全了置换过程中丢失的位。最后,基于 DES 子密钥的高熵特性,提出了熵阈值过滤策略,在保证误检率为 0 的同时,保持低漏检率。实验结果表明,该算法在内存取证场

www.c-10czz.com

景下能够准确、高效地识别和提取 DES 密钥，显著提升了加密数据取证的效率与完整性。

该算法在特定场景下展现出优异性能，但受限于设计逻辑，它的应用场景仍存在一定边界。在加密库适配方面，算法的反混淆策略基于 OpenSSL 库的 DES 密钥混淆机制设计，若目标系统采用 Crypto++、Libgcrypt 等其他加密库，因不同库的密钥混淆逻辑与数据存储格式存在差异，算法将无法有效完成子密钥还原。为突破现有局限，进一步提升算法的适用性与性能，未来可针对主流加密库的 DES 密钥混淆机制开展系统性研究，梳理不同库的混淆逻辑差异并建立统一的反混淆规则库，通过动态识别目标系统使用的加密库类型，自动匹配对应的反混淆策略，实现跨库 DES 密钥搜索。

参考文献(References)

[1] Volatility Foundation. Volatility 3 technical reference manual [EB/OL]. (2025-05-20)[2025-10-30]. https://volatility-foundation.org.

[2] Sadek I, Chong P, Rehman S U, et al. Memory snapshot dataset of a compromised host with malware using obfuscation evasion techniques[J]. Data in Brief, 2019, 26: 104437.

[3] Dolan-Gavitt B, Leek T, Hodosh J, et al. Tappan zee (north) bridge: mining memory accesses for introspection [C/OL]// CCS, 2013: 839-850. https: // dl. acm. org/doi/10. 1145/2508859.2516661.

[4] Taubmann B, Frädrich C, Dusold D, et al. TLSkex: harnessing virtual machine introspection for decrypting TLS communication[J].Digital Investigation, 2016, 16: S114-S123.

[5] Taubmann B, Alabduljaleel O, Reiser H. DroidKex: fast extraction of ephemeral TLS keys from the memory of Android apps [J]. Digital Investigation, 2018, 26 (S): 67-76.

[6] Halderman J, Schoen S, Heninger N, et al. Lest we remember: cold-boot attacks on encryption keys[J]. Communications of the ACM, 2009, 52 (5): 91-98.

[7] Moriconi F, Levillain O, Francillon A, et al. X-Ray-TLS: transparent decryption of TLS sessions by extracting session keys from memory [C/OL]// ASIACCS, 2024: 35-48. https://dl.acm.org/doi/10.1145/3634737.3637654.

[8] Singh G, Supriya S. A study of encryption algorithms (RSA, DES, 3DES and AES) for information security[J]. International Journal of Computer Applications, 2013, 67(19): 33-38.

[9] Mishra D, Kulkarni P. A survey of memory management techniques in virtualized systems[J]. Computer Science Review, 2018, 29: 56-73.

[10] Ottmann J, Breitinger F, Freiling F. Defining atomicity (and integrity) for snapshots of storage in forensic computing[C/OL]// DFRWS EU, 2022. https: // www.dfrws.org/conferences/dfrws-eu-2022.

[11] 潘吴斌，程光，郭晓军，等.基于信息熵的自适应网络流概念漂移分类方法[J]. 计算机学报，2017, 40 (7): 1556-1571.

作者简介：

黄景昊(2001-)，男(汉族)，漳州，硕士生。研究领域：网络安全。

吴礼发(1968-)，男(汉族)，蕲春，教授，博士。研究领域：软件安全、网络协议逆向分析、网络安全。本文通信作者。

(上接第 66 页)

[4] 朱强军，胡斌.基于 IEMA 和混合小目标的 YOLOv8s 交通标志检测方法[J].湖北民族大学学报(自然科学版)，2025，43(2)：202-209.

[5] 孙铭见，刘国洋，李春伟.基于改进 YOLOv8s 算法的饮品瓶盖缺陷检测[J].包装与食品机械，2025，43(2)：20-29.

[6] 胡成峰，黄琳涵，郑权，等.RT-TLTP 番茄叶片病虫害检测方法研究[J].计算机工程与应用，2026，62(6)：253-266.

[7] 王军，万书东，程勇.基于改进 YOLOv5s 的白酒瓶盖瑕疵检测[J].包装工程，2024，45(7)：180-188.

[8] 贾岚，徐雪环，罗德弢，等.基于 CrossStage-YOLO 的跨阶段作物害虫检测[J].南京农业大学学报，2025，48(3)：744-754.

[9] Yu Y, Li H, Shi H, et al. Question-guided feature pyramid network for medical visual question answering[J]. Expert Systems with Applications, 2023, 214: 119148.

[10] Qin L, Pang W, Zhao D. A feature pyramid network with adaptive fusion strategy and enhanced semantic information[J]. Multimedia Systems, 2024, 30(4): 171.

[11] Chen Z, Zhang Y, Wang Y, et al. Robust image inpainting forensics by using an attention-based feature pyramid network[J]. Applied Sciences, 2023, 13(16): 9196.

[12] Sisniega A, Stayman J W, Capostagno S, et al. Accelerated 3D image reconstruction with a morphological pyramid and noise-power convergence criterion[J]. Physics in Medicine & Biology, 2021, 66(5): 055012.

[13] Eickhoff K, Hillebrand A, De Jong M C, et al. Population receptive field models capture event-related MEG responses[J]. Journal of Vision, 2024, 24(10): 647.

[14] Wu Y, Liu J, Gong M, et al. Joint semantic segmentation using representations of LiDAR point clouds and camera images[J]. Information Fusion, 2024, 108: 102370.

[15] Song L, Shang X, Zhou R, et al. A multi-group multi-stream attribute attention network for fine-grained zero-shot learning[J]. Neural Networks, 2024, 179: 106558.

[16] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[C/OL] // ICCV, 2017: 2980-2988. https://ieeexplore.ieee.org/document/8237586.

作者简介：

彭来湖(1980-)，男(汉族)，温州，教授，博士。研究领域：机电一体化及智能制造。本文通信作者。

梁铭宇(2001-)，男(汉族)，杭州，硕士生。研究领域：机电一体化，机器学习。

陈　秀(1984-)，女(汉族)，温州，工程师，学士。研究领域：机器设计自动化装备。

徐步都(1992-)，男(汉族)，温州，高级工程师，学士。研究领域：机器设计自动化装备。

www.rjgcz.com