基于结构链逆向的内存碎片文件雕刻算法

李炳龙，周振宇，张宇，张和禹，常朝稳
（信息工程大学密码工程学院，河南 郑州 450001）

摘 要：为解决内存映像中碎片证据文件提取问题，针对 doc、pdf 等常见文件类型，提出了一种基于内存映像的碎片文件雕刻模型。基于该模型，设计了基于文件对象结构链逆向的碎片文件雕刻算法，能够获取遗留在内存中的文件数据。实验结果表明，该算法能够成功从内存映像中雕刻出文件相关的元数据信息，例如文件名、文件来源及操作行为等，雕刻精确度达到 100%；而且在典型应用情况下，文件内容数据雕刻精度达到 87.5%，远高于基于磁盘文件雕刻算法的精确度。

关键词：文件雕刻；内存取证；内存碎片；碎片连接；结构逆向
中图分类号：TP309
文献标识码：A
DOI: 10.11959/j.issn.1000−436x.2021143

Memory fragment file carving algorithm based on the reverse of the structure chain

LI Binglong, ZHOU Zhenyu, ZHANG Yu, ZHANG Heyu, CHANG Chaowen
Cryptography Engineering Academy, Information Engineering University, Zhengzhou 450001, China

Abstract: To address the extraction of document evidence for doc, pdf, and other common file types in the memory image, the model of fragment file carving based on memory image was proposed. Then, on the basis of the model, the fragment file carving algorithm based on the reverse of file object structure chain was designed and implemented, the algorithm was able to get file data left behind in the memory image file. The experimental results show that the proposed algorithm can successfully carve out of memory file's metadata, and the accuracy is 100%, and in a typical application case, the accuracy of the algorithm for memory file can achieve 87.5%, far higher than disk-based file caving algorithm.

Keywords: file carving, memory forensics, memory fragment, fragment adjacent, structure reverse

1 引言

随着信息技术的迅猛发展，网络犯罪事件频繁发生，例如电信诈骗、信息勒索以及 APT（advanced persistent threat）攻击等。数字取证调查已经成为制止和威慑网络犯罪的关键技术手段之一[1]。磁盘取证是数字取证领域中一种重要的调查技术，对于数字犯罪调查具有重要意义。但随着反取证技术的发展，新型“无文件攻击”[2]等网络威胁仅在内存中运行，不在磁盘上留下任何痕迹信息，具有更强的隐蔽性和复杂性。此外，磁盘容量的不断增大导致磁盘取证具有局限性。内存中含有与网络攻击威胁有关的大量实时证据和线索，而且内存中的网络连接、系统加载的模块以及执行的指令等证据信息和磁盘中的证据不同，更能说明网络攻击威胁发生的场景。因此内存取证成为当前数字调查领域的重要研究方向[3-5]。

目前，内存取证主要聚焦于基于内存映像的进程、网络连接信息、剪贴板数据、命令行历史、口令等证据的提取和分析[4-7]，而针对内存映像中数据文件雕刻技术研究较少。在早期内存取证实践工作中，调查人员利用 strings、WinHex 等工具[8]，从内

收稿日期：2020−12−03；**修回日期**：2021−03−01
基金项目：国家自然科学基金资助项目（No.60903220）
Foundation Item: The National Natural Science Foundation of China (No.60903220)

存映像中搜索口令、IP 地址、Email 地址等证据信息，该方法的前提是必须知道所要查找信息的内容。之后，有研究深入讨论和分析了基于内存映像精确提取证据的可行性[9-11]。Kornblum[12]提出一种内存映像中进程对应的可执行文件的提取方法，通过使用指针来重构可执行文件，并分析了由于可执行文件运行时的多种变化因素，导致提取的可执行文件与其在磁盘上相应的文件不完全相同的问题。此外，Dolan-gavitt[13]利用虚拟地址描述符（VAD, virtual address descriptor）树来定位、解析结构，遍历进程的虚拟内存空间，并为取证调查人员提供内存转储中有用的信息。Van-baar 等[14]进一步提出了基于 VAD 的内存数据文件提取方法，即通过遍历 VAD 树定位共享文件，然后查找对象表，进而找到文件信息。以上 2 种方法具有局限性，即如果进程关闭或者内存中进程结构信息被覆盖将不能有效定位 VAD 信息，从而造成文件信息不能恢复。有学者提出了基于页面哈希比较的数据文件提取方法[15]，该方法计算内存映像中页面哈希，并和磁盘中文件对应页的哈希进行比较，提取内存映像中的文件碎片，进而重建文件内容，要求取证调查人员必须知道需要重建文件的内容，而现实网络攻击调查中调查人员并不了解内存中的数据文件内容。此外，当磁盘中文件映射到内存页面时，内存中文件数据有可能被修改，导致内存页面哈希值不同于磁盘上对应页面，从而造成内存文件提取无效。Gao 等[16]研究了实时系统中 QQ 信息取证方法，从中获取通信列表、QQ 账号、聊天记录、QQ 讨论组、显示名称，该方法通过逆向分析 QQ 的内存结构来实现，但该方法仅适用于 QQ 应用程序，不具有通用性。Volatility 内存取证框架和 FATKit 内存取证分析工具套件能够从易失性内存映像中提取取证痕迹，但更多集中于内存映像中进程、线程等结构，以及可执行文件提取[17]。此外，虽然基于磁盘的文件雕刻算法有很多，例如 Foremost、Scalpel、Test_disk、EnCase 等[18-20]，但是应用这些算法的实验结果表明，生成的文件雕刻结果精确度极低。综上所述，基于内存映像的碎片数据文件雕刻算法的通用性问题没有得到很好的研究。为了解决基于内存映像的碎片文件雕刻算法的通用性问题，探索物理内存中新的文件雕刻机制具有非常重要的理论意义和现实价值[21-23]。

本文主要研究工作如下。

1) 建立了基于内存映像的碎片文件雕刻模型。从集合论的角度，分析了内存映像碎片集合中元素的特性。从内存文件构成原理，给出了内存文件的形式化表达方式，提出了内存碎片文件雕刻问题可以抽象为内存碎片集合到内存文件的映射问题，设计了内存碎片文件雕刻模型。

2) 设计了基于文件对象结构链逆向的内存碎片文件雕刻算法。利用十六进制编辑器分析了内存映像中文件对象及其相关结构的特征字段，设计了基于结构链逆向的内存碎片文件雕刻算法，解决了内存文件碎片子集中元素确定子问题、文件碎片子集中元素连接关系顺序子问题，以及文件元数据构建子问题。

3) 实验结果表明，基于结构链逆向的内存碎片文件雕刻算法通用性强，不但能够雕刻内存中文件内容信息，而且能够雕刻文件元数据信息，例如文件来源、文件名称等，还适用于应用进程关闭时的文件雕刻情况。实验分析进一步表明，通过对雕刻的碎片文件的内部结构进行深度分析，能够找到网络攻击过程中病毒木马的感染过程，这对于实时取证调查具有重要意义。

2 相关工作

内存取证目前是数字取证领域重点研究方向之一[6-7]。早期内存取证研究进展较慢，为此 2005 年 DFRWS（Digital Forensic Research Workshop）发布了物理内存取证分析挑战，鼓励研究人员开展内存取证技术研究[24]。内存取证分为物理内存获取和物理内存分析 2 个阶段。内存获取分为基于硬件和基于软件的 2 种方法。目前，基于硬件的内存获取典型工具是基于 PCI 扩展机制的 Tribble 工具[9]。该方法优点是在获取内存数据时不会造成内存数据的改变，但是其局限性在于必须在取证事件发生之前安装到被感染机器中。基于软件的内存获取方法则较普遍，可以根据事件响应的需要灵活安装。Guidance 公司发布的 EnCase 6.11 以上取证工具套件中含有 WinEn 物理内存映像工具，并且支持 Windows 32 位和 64 位操作系统，能够生成 3 种不同压缩级别的内存映像[20]。ManTechs 公司发布了 MDD 内存获取工具，该工具根据 GPL（GNU general public license）免费发布，能够获取物理内存映像，并且以原始内存格式进行转储。Win32dd 是一个完全开源的内存映像获取工具[25]，该工具基于

Windows 内核设计实现，适用于 Windows2003 或 Vista 系统的内存获取，获取的内存映像是原始格式。

内存取证分析技术研究目前主要集中于进程注册表等内核结构逆向分析重建。2005 年，DFRWS 的内存取证挑战产生了学术界认为开创性的 3 个研究成果[24]：一是 Mariusz 提出的进程和模块枚举方法；二是 Chris 等设计的 memparser 工具；三是 George 等设计的 Kntlist 工具。这 3 个工具基本原理是逆向分析对应版本的 Windows 进程结构，在内存映像文件中搜索全局变量，雕刻重建进程及其双向链表。这些方法局限性在于不能雕刻内存映像中不在双向链表中的进程等对象。为此，Schuster[24] 通过分析内存中进程和线程结构，提出了一种基于搜索模式特征的进程和线程雕刻算法，通过扫描内存映像文件，定位并重建相应的对象，能够雕刻由于 DKOM（direct kernel object manipulation）攻击隐藏和已终止的进程和线程。有学者通过逆向分析注册表的内存分配单元，设计并实现了基于池签名特征的注册表雕刻工具[22]。以上算法和工具仅能支持少量 Windows 系统版本的内存映像。Volatility 是数字取证领域经典的内存取证分析框架，应用了 Schuster 的基于签名特征扫描的对象雕刻技术[17]。为了支持更多 Windows 操作系统版本的内存映像，Volatility 引入了基于内核模板的机制，即在该框架集成多个 Windows 操作系统主版本的内核结构信息，从而改善不同 Windows 操作系统版本的内存映像数据雕刻。然而，该方法并未考虑 Windows 操作系统次版本变化（例如同一主版本因安全补丁更新等因素产生的不同次版本）而导致操作系统内核结构模板的改变，从而造成内存映像雕刻信息错误或者遗漏。而且基于签名特征扫描技术复杂且易发生错误[26]。Okolica 等[27]利用嵌入内存映像中的调试结构和程序数据库（PDB, program database）构建了一个 Windows 操作系统不可知的内存映像数据雕刻算法，该算法支持任意 Windows 操作系统内存映像，能够提取出进程、注册表以及网络活动等信息。

上述研究在内存进程重建、注册表逆向分析等方面取得了不同研究进展，并且也有针对内存映射文件提取方法进行的研究，而针对内存映像中数据文件雕刻技术研究较少，这也是本文重点要解决的问题。

3　内存碎片数据文件雕刻算法模型

3.1　问题描述

内存数据文件是 doc、pdf、xls、txt 以及 jpg 等类型文件在内存中由用户进程或攻击进程打开和访问的文件。内存文件由两部分构成，分别为文件内容信息和文件名（文件路径）等元数据信息。这些信息含有与网络犯罪有关的重要证据。内存数据文件的存储特性与磁盘中文件的存储状态和规律不同，磁盘中文件构成单元（簇）多是连续存放，而内存文件构成单元（内存页面）多是不连续存放，这导致内存文件构成单元严重碎片化。

本文假设在 t 时刻获取内存映像，构成内存映像的页面称之为“碎片”，那么内存映像可以抽象为一个碎片集合 $S=\{f_1,f_2,\cdots,f_i,\cdots,f_j,\cdots,f_n\}$，其中，$f_i$ 表示内存映像中的任意一个内存碎片，n 表示集合 S 的大小，并且 n 值取决于内存介质容量以及内存页面的大小（在内存容量一定的情况下，页面越大，则内存碎片集合 S 中元素个数就越小，通常情况下内存页面大小为 4 KB，即 4 096 B）。集合 S 具有如下特性。

1) 确定性。对于 $1\leqslant i\leqslant n$，f_i 表示集合 S 中的内存碎片元素，用 $f_i\in S$ 表示，这表明 f_i 是在 t 时刻获取的内存映像中确定的一个元素，而不是其他时刻获得的内存映像中的元素。

2) 互异性。对于 $1\leqslant i,j\leqslant n$，$f_i,f_j$ 表示集合 S 中的内存碎片，如果 $i\neq j$，那么 $f_i\cap f_j=\varphi$，表示碎片集合 S 中 2 个不同碎片在内存空间上不存在交集，并且 2 个碎片的内容及其元数据（例如内存碎片的地址信息等）都不相同，即集合中的任何 2 个元素都不相同，或者在同一集合里不能出现相同的元素。

3) 无序性。集合中的元素是平等的，没有先后顺序。因此判断 2 个集合是否相同，只需要比较它们的元素是否一样，不需要考察排列顺序是否一样。但由于内存数据的动态变化性，以及内存映像获取的时间变化性，使不同时刻的内存碎片集合都是不同的。

根据操作系统运行原理，内存文件由多个页面构成。由内存映像碎片集合 S 中元素特性可知，内存文件是 S 的一个子集，即文件碎片子集，用 file_i 表示，其中 $0\leqslant i\leqslant k$，k 表示内存碎片中最大文件个

数。此外，由于内存文件的页面碎片之间存在一定的顺序关系，如果内存文件的页面碎片之间的顺序破坏，则导致内存文件损坏或者内容变化。为此，本文将内存文件抽象为一个序列，用 $(\text{file}_{i,k})$ 表示。

$$(\text{file}_{i,k})=(f_{i,1},f_{i,2},\cdots,f_{i,j},f_{i,j+1},\cdots,f_{i,k})$$

其中，$1\leqslant j\leqslant m$，m 表示 file_i 碎片子集中元素个数。$(\text{file}_{i,k})$ 具有如下特性。

1) 有限性，即 $(\text{file}_{i,k})$ 中的项是有限的，这是根据文件构成原理确定的，尽管理论上文件长度可以无限，但不具有实际意义。

2) 有序性，即 $(\text{file}_{i,k})$ 中项与项之间的关系是有序的，这种有序构成了文件的内在关系，具体可以体现为内存页面间的结构关系、语义关系、签名特征关系。例如，$f_{i,1}$ 是该序列中的第一项，表示文件头碎片；$f_{i,k}$ 是该序列最后一项，表示文件尾碎片，这通常是根据文件类型头尾签名特征确定的。

3) 元数据性，用来说明 $(\text{file}_{i,k})$ 的文件名、大小，甚至是由哪个进程打开等信息。需要说明的是，$(\text{file}_{i,k})$ 具有元数据，但并不是基于该序列中的页面碎片提取，而是从含有操作系统相关结构的页面碎片中提取。

为此，一个内存碎片文件可以抽象为 $\text{metadata}\,\|\,(\text{file}_{i,k})$。根据以上分析，基于内存映像的碎片文件雕刻问题可以看作集合 S 到 $\text{metadata}\,\|\,(\text{file}_{i,k})$ 的一个映射，即 $S\to\text{metadata}\,\|\,(\text{file}_{i,k})$，也就是找到这种映射关系，就能解决内存碎片文件雕刻。

3.2 碎片雕刻问题分析

内存碎片文件雕刻问题是一个内存碎片集合到文件元数据和文件序列的映射问题，该问题可以分解为以下 3 个子问题。

1) 文件碎片子集元素确定子问题

含有数据文件的页面碎片集合仅是整个内存映像碎片集合的一部分。此外，含有操作系统内核模块、内核进程、硬件驱动，以及应用进程等的页面也都是内存映像碎片集合中的元素。因此构成数据文件的碎片集合是内存映像碎片集合的一个子集，利用集合划分思想，数据文件碎片子集和内存碎片集合 S 之间具有如下关系

$$S\supset\text{file}_1\cup\text{file}_2\cup\cdots\cup\text{file}_i\cup\cdots\cup\text{file}_k$$

其中，$0\leqslant i\leqslant k$，file_i 表示文件碎片子集，其实质是具有特定文件类型（如 office 文件类型、Acrobat pdf 文件类型等）的一个数据文件的所有页面。文件碎片子集元素确定子问题就是确定一个数据文件中的所有页面碎片，即建立碎片集合 S 到文件碎片子集 file_i 的映射，即 $S\to\text{file}_i$，利用该映射关系获得数据文件碎片子集中的所有碎片。

2) 文件碎片子集元素关系顺序确定子问题

由于内存碎片集合 S 中元素的无序性，并且文件碎片子集 $\text{file}_i\subset S$，根据集合中元素无序性特征，则 file_i 中元素也具有无序性。假设 file_i 中有 m 个元素碎片，则可能的连接顺序共有 $m\times(m-1)\times(m-2)\times\cdots\times2\times1$ 种。文件碎片子集元素关系确定子问题就是要找到 file_i 中元素之间的唯一的序列，即 $\text{file}_i\to(\text{file}_{i,k})$ 的映射。

3) 文件元数据构建子问题

文件元数据通常是指文件系统中维护文件内容数据的相关数据。目前，内存数据文件元数据尚没有定义，本文认为内存文件元数据是指在内存中由操作系统及其相关结构维护的文件信息，例如文件名、文件大小等数据。因此，含有内存文件元数据的页面碎片并不是某个 file_i 碎片子集中的元素，而是具有操作系统结构的内存碎片。要构建内存文件元数据，前提是必须找到含有操作系统相关数据结构的内存碎片，并在该内存碎片上找到文件元数据的具体位置。

由于内存碎片元素确定性、互异性和无序性，通过直接分析内存碎片中的二进制数据难以解决上述 3 个子问题。基于内存管理理论和机制，内存页面是通过操作系统的内核对象结构进行管理的，也就是说如果找到内存映像中含有操作系统相关结构的碎片，并针对文件对象及其链接关系进行逆向重建，有望解决碎片文件雕刻的 3 个子问题。

3.3 内存碎片文件雕刻模型

碎片文件雕刻模型的基本思想是利用内存中操作系统结构逆向分析技术，分析 Windows 操作系统中文件对象结构及相关结构，逆向构建文件对象的结构链，通过结构链中的指针关系确定文件碎片子集中的元素及碎片元素间的连接关系，并利用相关结构中文件名等字段构建文件元数据。本文选择将文件对象作为结构链逆向重建的起始结构，其优点在于能够解决现有雕刻算法中进程被关闭或者

被覆盖后造成文件不能雕刻的问题。利用 WinHex 工具分析可知，文件对象结构的内存分配单元具有明显的签名特征。文件对象分配单元签名十六进制特征如图 1 所示。

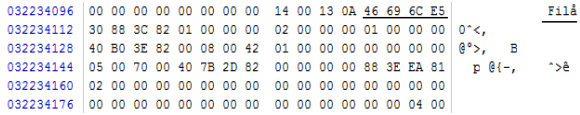

图 1　文件对象分配单元签名十六进制特征

每个文件对象的池分配单元的签名特征都含有“46 69 6C E5”这样的十六进制信息，因此该签名特征可以定位文件对象结构链的起始结构。此外，构成文件的内存页面都有一个页面指针，通过文件页面指针就能够确定文件的内存页面以及该页面在文件中的位置顺序。本文通过重建文件对象结构及其中的关键字段，并根据字段中的指针地址确定文件对象结构链接指向关系，最终确定文件的碎片元素及其关系，构建的内存碎片文件雕刻模型如图 2 所示。

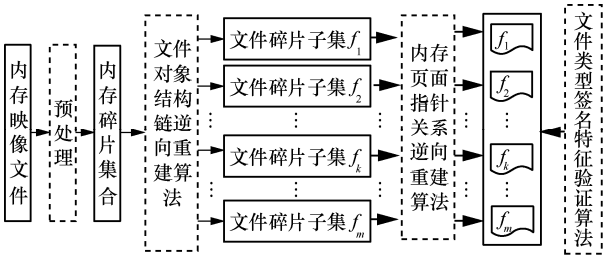

图 2　内存碎片文件雕刻模型

如图 2 所示，内存碎片雕刻过程可分为 4 个阶段：预处理、文件对象结构链逆向重建算法、内存页面指针关系逆向重建算法和文件类型签名特征验证算法。

预处理。先对内存映像进行扫描分析，去除非数据文件内存碎片，包括利用 0/1 二进制数据统计特征过滤不含有任何数据的内存页面碎片，利用内存取证中的进程重建机制去除含有可执行代码的内存碎片页面。最终获取内存数据文件碎片及元数据碎片的集合。

文件对象结构链逆向重建算法。1) 利用文件内核对象内存分配签名特征，扫描内存映像碎片集合，确定该签名特征位置，同时记录内存映像碎片集合中签名个数（对应内存中文件个数）；2) 根据签名特征位置，定位文件对象分配单元，并逆向重建其关键字段；3) 根据文件对象结构的链接指向关系，确定指向内存碎片的页面指针。依据页面指针确定文件碎片子集中元素的个数。

内存页面指针关系逆向重建算法。确定页面指针的关系，构建文件碎片子集中页面之间的连接顺序关系。根据内存碎片文件雕刻模型可知，碎片文件雕刻成功的关键是能够逆向重建文件对象，并且能够获得结构中的关键变量，尤其是结构链指针。如果指针值有误差，则可能导致错误的雕刻结果。为了进一步提升雕刻结果的正确性，引入文件类型签名特征验证算法针对雕刻结果进行验证，从而确认雕刻结果十六进制类型是否符合文件类型要求。

文件类型签名特征验证算法。利用文件类型十六进制头尾特征，例如 jpg 文件类型的文件头十六进制特征是“FFD8FFE000104A464946”，文件尾特征是“FFD9”，针对内存中 jpg 雕刻文件进行比较验证，从而验证雕刻文件的签名特征。

4　算法设计

4.1　文件对象相关结构逆向重建

Windows 系统的内核结构、内存的相关管理机制并没有完全公开，并且不同系统版本、机器字长、分页模式等因素对内存映像分析影响很大。本文使用微软公司提供的内核调试工具 WinDbg 对文件对象及相关结构进行逆向分析重建。文件对象及相关结构在内存中的结构链关系如图 3 所示。

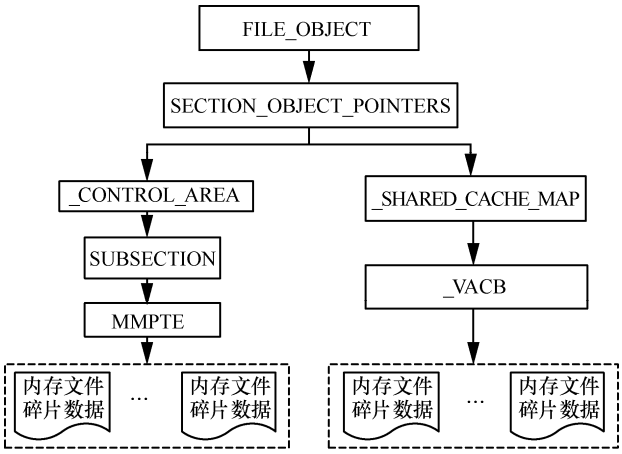

图 3　内存文件结构链关系

通过逆向重建图 3 中文件对象及其链接关系，就能确定内存文件碎片集合及其之间的关系，需要逆向重建的结构如下。

1) FILE_OBJECT

Windows 文件对象（FILE_OBJECT）是 Windows 系统运行中进程访问（修改、删除）文件

时构建的一个结构，并被分配给那个文件。重建文件内容数据，首先要定位 FILE_OBJECT。该结构的关键变量如下。

```
lkd> dt nt!_FILE_OBJECT
    +0x000 Type
    +0x004 DeviceObject
    +0x014 SectionObjectPointer
    +0x030 FileName
```

FILE_OBJECT 结构包含几个重要的成员，通过偏移量 0x030 处的 FileName 可以查看文件的名称；0x004 处的 DeviceObject 域包含一个指向 DeviceObject 的指针，其中包含了驱动和设备信息；对于数据文件雕刻来说，最重要的成员变量在偏移量为 0x014 的位置，它是一个指向 SECTION_OBJECT_POINTERS 结构的指针，下文详细分析了该结构。

2) SECTION_OBJECT_POINTERS

内存管理器和缓存管理器应用该结构存储文件映射和缓存有关的信息，该结构的关键变量定义如下。

```
lkd> dt nt! _SECTION_OBJECT_POINTERS
    +0x000 DataSectionObject
    +0x004 SharedCacheMap
    +0x008 ImageSectionObject
```

该结构包含 3 个指针变量。其中，DataSectionObject 指向 CONTROL_AREA 结构，用于维护数据文件，例如 Microsoft word 文件；ImageSectionObject 用来表示内存中可执行文件；SharedCacheMap 指向一个 SHARED_CACHE_MAP 结构变量，该结构与操作系统中的高速缓存有关。

3) _CONTROL_AREA

_CONTROL_AREA 是一个内存管理结构，是整个内存映射的核心，同时也是联系 SUBSECTION 和 MMPTE 结构的重要结构。该结构关键变量如下。

```
lkd> dt nt! _CONTROL_AREA
    +0x000 Segment
    +0x01c Subsection
    +0x040 FilePointer
```

偏移量 0x01c 的变量 Subsection 是指针变量，通过该指针可以定位 SUBSECTION 结构。

4) SUBSECTION 结构

SUBSECTION 结构用来管理文件映射到内存中的各个页面。该结构关键变量如下。

```
lkd> dt nt!_SUBSECTION
    +0x000 ControlArea
    +0x010 SubsectionBase
    +0x01c NextSubsection
```

偏移量 0x010 处的 SubsectionBase 是一个指向 MMPTE 结构的指针，这个结构实际上是一个原型 PTE 的数组。原型 PTE 与硬件 PTE 相关联，因此通过该结构可以找到数据文件页面的存储内容。MMPTE 结构变量如下。

```
lkd>dt_MMPTE -r1
nt!_MMPTE
    +0x000 u: __unnamed
        +0x000 Long: Uint8B
        +0x000 HighLow: _MMPTE_ HIGHLOW
        +0x000 Hard: _MMPTE_HARDWARE
        +0x000 Flush: _HARDWARE_PTE
        +0x000 Proto: _MMPTE_PROTOTYPE
        +0x000 Soft: _MMPTE_SOFTWARE
        +0x000 Trans: _MMPTE_TRANSITION
        +0x000 Subsect: _MMPTE_ SUBSECTION
        +0x000 List: _MMPTE_LIST
```

5) _SHARED_CACHE_MAP

_SHARED_CACHE_MAP 结构管理内存中的缓存文件，该结构的关键变量如下。

```
lkd> dt nt! _SHARED_CACHE_MAP
    +0x000 NodeTypeCode
    +0x002 NodeByteSize
    +0x008 FileSize
    +0x028 ValidDataLength
    +0x030 ValidDataGoal
    +0x038 InitialVacbs[4]
    +0x058 Vacbs
```

该结构中的 Vacbs 指针变量指向_VACB 结构，使用的是虚拟地址，通过该地址能够找到数据文件在虚拟地址中的数据。尽管并不是文件的所有部分都被映射到缓存中，但是缓存中的文件部分数据具有极其重要的特性，因为如果文件的部分内容在缓存中被找到，则意味着文件的这部分内容已经被某一进程最近或经常使用和访问。

4.2　内存碎片文件雕刻算法

本节基于逆向工程重建的文件对象及其相关的内存结构设计内存碎片文件雕刻算法。如图 4 所示，内存碎片文件雕刻算法分为内核文件对象扫描重建、文件碎片提取与重建、缓存碎片文件雕刻 3 个子算法。该算法中需要多次定位对应的结构，根据内存页式管理的特点，虚实地址转化首先要定位页目录表的基地址 DTB。分析得知，文件对象重建中虚拟地址的转化需要 csrss.exe 系统进程的 DTB，进而按照 32 位或 64 位系统虚拟地址字段的划分来完成整个的地址转化过程，定位文件典型及其相关结构键的物理地址。

1) 内核文件对象扫描重建子算法的详细过程如下。

① 利用文件对象签名特征扫描算法识别文件对象分配单元，并定位 FILE_OBJECT 对象，重建该结构的“SectionObjectPointer”字段变量，该字段指向“SECTION_OBJECT_POINTERS”结构，该结构由 Windows 构建。

② 定位 SectionObjecPointer 指向的 SECTION_OBJECT_POINTERS 结构，重建该结构的 DataSectionObject 和 SharedCacheMap 字段变量。

2) 文件碎片提取与重建子算法

① 定位 DataSectionObject 指向的 CONTROLAREA 结构，重建该结构中的 Subsection 字段变量。

② 定位 Subsection 字段，并重建该结构中的

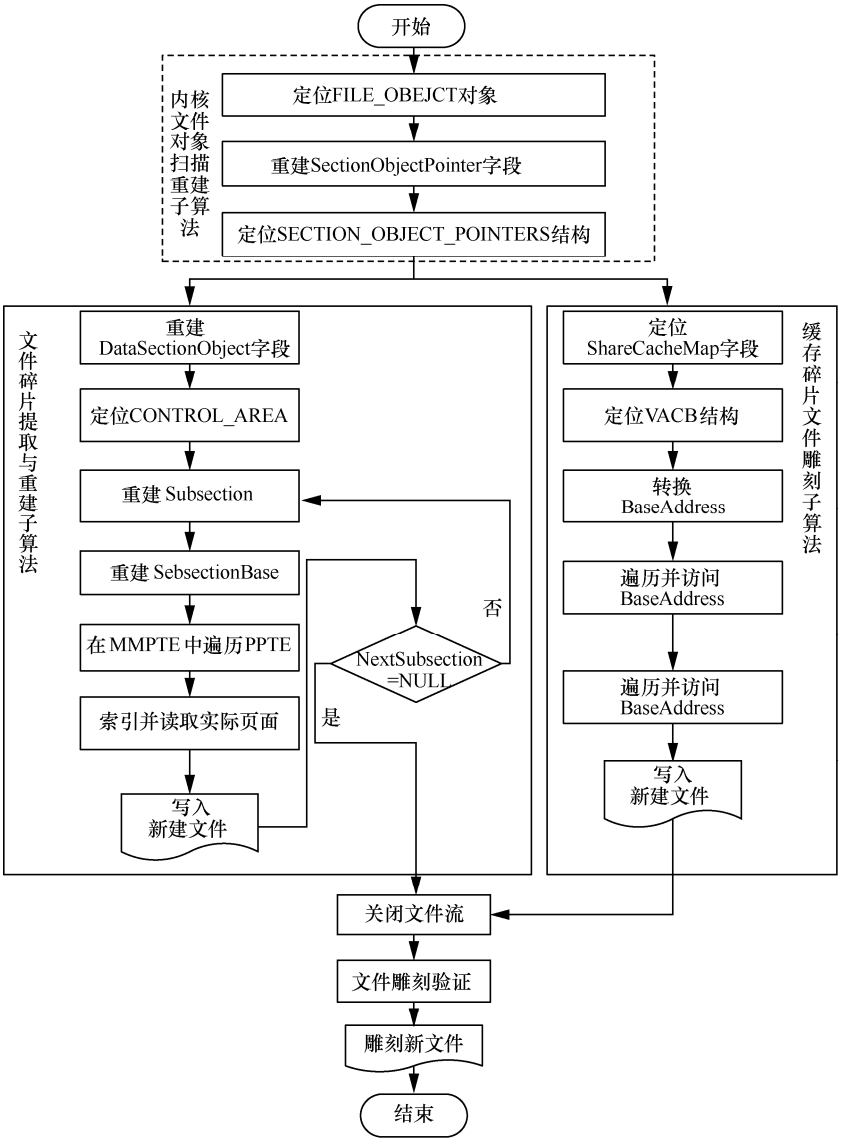

图 4　内存碎片文件雕刻算法

SubsectionBase 的指针和 NextSubsection 的指针变量。

③ 定位 SubsectionBase 地址，重建 MMPTE 数组结构变量。

④ 遍历 MMPTE 数组指向的页面，并将其数据写入到新建文件中。每个 MMPTE 可索引一个 4 096 B 的页面，而一个完整的扇区有 512 B，每个 MMPTE 最多可以对应 8 个扇区。

此外，每一个 Subsetction 均有一个指针指向 NextSubsetction，表示文件下一部分所处的内存页面。如果文件的大小足以用一个 SubSetction 来管理，则此处为空。

3) 缓存碎片文件雕刻子算法

① 定位 SharedCacheMap 字段指向的 SHARED_CACHE_MAP 结构，重建该结构的 Vacbs 指针变量。

② 定位 VACB 结构，重建 VACB 结构。转换 VACB 结构中的 BaseAddress 地址为物理地址。

③ 如果 VACB 结构中的 BaseAddress 为 0，则表示该文件对应的缓存无效，算法退出。否则继续执行④。

④ 定位 VACB 结构中的 BaseAddress 变量，遍历该结构数组，将每一个页面数据写入新建文件，重建碎片文件内容。

5 实验结果及分析

本文以 Windows 7 操作系统为内存文件雕刻实验对象。根据 StatCounter 在 2020 年 10 月发布的全球 PC 操作系统市场份额排名，Windows 7 操作系统占据 16.8%的市场份额[28]，Windows 7 操作系统的实际用户数量肯定还要高于这个比例。虽然不同版本操作系统的内存对象结构不尽相同，但是本文中介绍的获取物理内存中数据的基本方法可以推广到 Windows 8、Windows 10 等操作系统中。

5.1 碎片文件雕刻结果

实验机器是 ThinkPad X230，Intel® Core™ i5-3230M CPU @2.60 GHz，内存容量为 4 GB，64 位 Windows 7 操作系统。选择 doc、pdf、txt、jpg 这 4 种常用文件类型数据文件进行测试，验证本文方法对内存中数据文件的元数据（如文件名）和文件内容的雕刻能力，各选择 10 个文件，文件详细情况如表 1 所示。

本文设计了 3 个不同的实验，用于验证本文方法在网络攻击的不同阶段（例如攻击前、攻击中、攻击后）物理内存中数据文件雕刻能力和可行性，实验过程分别如下。

表 1 不同类型文件尺寸

文件	doc/B	pdf/B	txt/B	jpg/B
f0	24 064	51 618	32	14 717
f1	26 112	66 067	698	60 019
f2	26 624	86 311	747	97 788
f3	27 136	138 276	778	122 854
f4	28 160	145 813	997	141 963
f5	39 424	159 331	3 917	226 174
f6	81 920	212 392	15 127	360 907
f7	118 784	272 669	24 154	421 069
f8	189 952	397 519	24 564	2 599 419
f9	233 984	2 173 636	318 997	6 523 649

1) 实验 1。从本地磁盘中分别利用美图看看、Microsoft Office、Acrobat 阅读器、Notepad.exe 等工具打开 f0 文件（注意文件后缀不同），然后转储内存映像。该实验主要用于测试内存映像中数据文件雕刻的可行性。

2) 实验 2。从本地磁盘中分别利用美图看看、Microsoft Office、Acrobat 阅读器、Notepad. exe 等工具打开表 1 中所有文件，然后转储内存映像。该实验主要验证攻击过程中黑客使用多个不同文件时内存映像中数据文件的雕刻能力，同时分析多个不同类型文件所占内存页面间是否相互影响。

3) 实验 3。关闭所有文件，然后转储内存映像。该实验验证黑客关闭攻击过程中打开使用的数据文件在内存中是否遗留文件痕迹。

针对实验 1 获得内存映像，雕刻结果从文件元数据和文件内容进行评价，文件元数据主要是指文件的名称，即文件的完整路径。Scalpel[29]是一款经典的基于磁盘的文件雕刻工具，本文选择 Scalpel 1.60 和内存碎片文件雕刻算法进行比较，结果如表 2 所示。

表 2 内存映像碎片文件雕刻结果

文件类型	本文方法		Scalpel 1.60	
	元数据	文件内容	元数据	文件内容
jpg	√	√	×	×
doc	√	√	×	×
pdf	√	√	×	×
txt	√	√	×	√×

表 2 中，√表示雕刻成功，×表示雕刻失败。从表 2 结果可知，本文方法不但能够雕刻出实验 1 中内存映像的文件内容数据，而且可以雕刻出文件的完整路径，这对于确定内存中文件来源具有意义，例如文件可能来源于网络。利用 Scalpel 1.60 工具针对该内存映像进行雕刻，只有 txt 文件可以进行成功雕刻，√×表示雕刻出 36 个 txt 文件，而 f0.txt 文件的内容分散在这些文件中，即虽然内容找到了，但没有成功雕刻一个完整文件。

针对实验 2 获得的内存映像，本文方法和 Scalpel 1.60 的实验结果如表 3 所示。

表 3　内存映像碎片文件雕刻结果（实验 2）

文件类型	本文方法		Scalpel 1.60	
	元数据/个	文件/个	元数据/个	文件/个
doc	10	8	0	0
pdf	10	8	0	0
txt	10	10	0	0
jpg	10	9	0	2
平均精确度	100%	87.5%	0	5%

从表 3 可以看出，本文方法能够雕刻出所有元数据，并能够雕刻出所有 txt 文件和大部分的 jpg、doc、pdf 文件。和实验 1 不同的是，雕刻失败的文件并不是较大的文件，即文件大小和文件成功雕刻之间没有关系，本文推测可能是由于多个文件同时在内存中打开而相互影响，有些文件页面交换到虚拟内存，导致内存中文件页面缺失。

针对实验 3 获得的内存映像，本文方法和 Scalpel 1.60 实验结果如表 4 所示。

表 4　内存映像碎片文件雕刻结果（实验 3）

文件类型	本文方法		Scalpel 1.60	
	元数据/个	文件/个	元数据/个	文件/个
doc	10	2	0	0
pdf	10	1	0	0
txt	10	10	0	0
jpg	10	4	0	1
平均精确度	100%	42.5%	0	2.5%

表 4 结果表明，实验 3 中文件内容雕刻精确度为 42.5%，远低于实验 2 中的 87.5%。可能原因是进程关闭后导致结构链逆向重建时某些关键指针字段值不能有效建立。另外，文件元数据也是通过缓存雕刻子算法获得的，这说明在实际的数字调查过程中需要综合利用文件雕刻算法获得有用的调查信息。

5.2　PDF 文件雕刻算法比较分析

文献[30]提出了一种利用 pdf 构成对象模式特征扫描和分类算法从内存映像中雕刻 pdf 文件的算法，该算法实验结果表明分层聚类算法优于 k-均值算法。本文利用文献[30]中 pdf 文件雕刻算法（其中分类算法利用分层聚类算法）针对实验 1、实验 2 和实验 3 获得的内存映像进行雕刻，雕刻实验结果如表 5 所示。

表 5　pdf 文件雕刻结果

实验	本文算法		文献[30]算法	
	元数据/个	文件/个	元数据/个	文件/个
实验 1	10	10	0	5
实验 2	10	8	0	4
实验 3	10	1	0	2
精确度	100%	63.3%	0	36.7%

表 5 实验数据表明，在所有实验场景中，本文算法成功雕刻 pdf 文件平均精确度为 63.3%，元数据精确度 100%，而文献[30]算法 pdf 文件平均精确度 36.7%，这些结果表明本文算法相比文献[30]算法能够雕刻更多 pdf 文件数据。另外，文献[30]算法对于 pdf 类型文件元数据精确度为 0，表明该算法不能从内存映像中雕刻 pdf 相关的元数据信息（例如文件名等）。

本文算法和文献[30]算法对于所有实验中的内存映像中的 pdf 文档内容的平均雕刻精度都未能达到 100%，可能原因在于构成 pdf 文件的对象元素有个别不在内存映像中。另外，本文雕刻算法平均精确度优于文献[30]算法，一方面，因为本文算法的结构链逆向分析方法能够有效确定单个 pdf 文件的对象元素及其连接关系，而文献[30]算法利用 pdf 对象模式特征确定 pdf 对象元素，导致不同 pdf 文档的相同对象元素的区分和重组难度大，尤其对于具有相似语义的文档来说则更容易混淆；另一方面，因为文献[30]方法仅能针对 pdf 文件，且仅关注 pdf 文件中的文本对象，实际上 pdf 文件中还包括其他的对象，例如图像等。本文算法可以针对不同文件类型，而且通过结构链之间的指针关系来确定构成文档内存页面之间的关系，通用性强。

5.3　宙斯木马病毒入侵分析

本案例选择 Honeynet 项目挑战提供的内存映

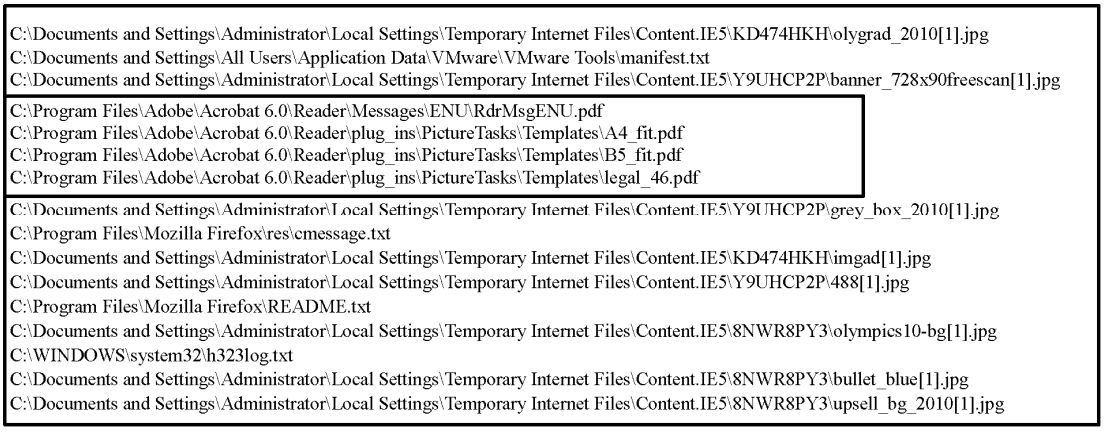

图 5　Bob.vmem 映像文件元数据雕刻

像文件 Bob.vmem[31]，该内存映像中可能存在 Zeus 病毒木马。Zeus 病毒木马通过 Acrobat 工具对系统进行感染，本文通过内存碎片文件雕刻算法对该内存映像中的 pdf 文件进行雕刻，并进行进一步分析，更加确认了 Zeus 病毒木马的感染过程。

利用本文方法对该内存映像进行雕刻，共雕刻出 16 个文件元数据，其中 pdf 文件 4 个，如图 5 所示。这些元数据表明，在 Zeus 病毒木马入侵期间黑客打开或者访问了这些文件。

利用 WinHex 工具针对雕刻出的 pdf 文件进行分析，发现其中含有“JavaScript”和“OpenAction”等 pdf 文件类型结构对象。综合该案例其他线索证据，黑客正是利用这些结构对象将 Zeus 病毒木马感染给该系统。

需要说明的是，Acrobat 阅读器不能打开雕刻出的 4 个 pdf 文件，但这并不影响通过这些 pdf 文件内容发现网络攻击的感染过程，这进一步表明内存数据文件的雕刻有助于网络攻击威胁调查和分析。

6　结束语

本文提出了一种基于内存映像的碎片文件雕刻模型，基于该模型提出一种结构链逆向的碎片文件雕刻算法，该算法能够获取遗留在内存中的文件数据。实验结果表明，本文算法能够成功雕刻恢复内存映像中的文件元数据信息，对于通常情况下的内存映像文件雕刻的精确度达到 87.5%，远高于基于磁盘文件雕刻算法（Scalpel 1.60），这对网络威胁攻击调查具有重要意义。在未来的工作中，将进一步探究 Linux 系统中碎片文件雕刻技术和算法。

参考文献：

[1] SERVIDA F, CASEY E. IoT forensic challenges and opportunities for digital traces[J]. Digital Investigation, 2019, 28: 22-29.

[2] SUDHAKAR, KUMAR S. An emerging threat Fileless malware: a survey and research challenges[J]. Cybersecurity, 2020, 3(1): 1-12.

[3] The Internet Crime Complaint Center. 2019 Internet crime report[R]. 2019.

[4] McAfee Labs. 2019 threats report[R]. 2019.

[5] CAVIGLIONE L, WENDZE S, MAZURCZKY W. The future of digital forensics: challenges and the road ahead[J]. IEEE Security & Privacy, 2017, 15(6):12-17.

[6] XIAO T, XU M, XU J. Acquisiting text documents opened by notepad from Windows7 RAM image[J]. Journal of Computational Information Systems, 2014, 10(16): 7117-7124.

[7] PATEL A, MISTRY N. An analyzing of different techniques and tools to recover data from volatile memory[J]. International Journal for Scientific Research & Development, 2013, 1(2):227-233.

[8] NUR A, MOHAMAD K, HASHEEM Y. Corrupted MP4 carving using MP4-Karver[J]. International Journal of Advanced Computer Science and Applications, 2016, 7(3):88-93.

[9] CARRIER B D, GRAND J. A hardware-based memory acquisition procedure for digital investigations[J]. Digital Investigation, 2004, 1(1): 50-60.

[10] MULLAN P, RIESS C, FREILING F. Forensic source identification using JPEG image headers: the case of smartphones[J]. Digital Investigation, 2019, 28: 68-76.

[11] BAHJAT A A, JONES J. Deleted file fragment dating by analysis of allocated neighbors[J]. Digital Investigation, 2019, 28: 60-67.

[12] KORNBLUM J D. Using every part of the buffalo in Windows memory analysis[J]. Digital Investigation, 2007, 4(1): 24-29.

[13] DOLAN-GAVITT B. The VAD tree: a process-eye view of physical memory[J]. Digital Investigation, 2007, 4: 62-64.

[14] VAN-BAAR R B, ALINK W, VAN-BALLEGOOIJ A R. Forensic memory analysis: files mapped in memory[J]. Digital Investigation, 2008, 5: 52-57.

[15] QUICK D, CHOO K K R. Impacts of increasing volume of digital forensic data: a survey and future research challenges[J]. Digital Investigation, 2014, 11(4): 273-294.

[16] GAO Y H, CAO T J. Memory forensics for QQ from a live system[J]. Journal of Computers, 2010, 5(4): 541-548.
[17] PETRONI N L, WALTERS A, FRASER T, et al. FATKit: a framework for the extraction and analysis of digital forensic data from volatile system memory[J]. Digital Investigation, 2006, 3(4): 197-210.
[18] 马庆杰, 李炳龙, 位丽娜. 基于 SQLite 内容雕刻的恢复技术[J]. 计算机应用, 2017, 37(2): 392-396.
MA Q J, LI B L, WEI L N. File recovery based on SQlite content carving[J]. Journal of Computer Applications, 2017, 37(2): 392-396.
[19] 高元照, 李炳龙, 陈性元. 基于 MapReduce 的 HDFS 数据窃取随机检测算法[J]. 通信学报, 2018, 39(10): 11-21.
GAO Y Z, LI B L, CHEN X Y. Stochastic algorithm for HDFS data theft detection based on MapReduce[J]. Journal on Communications, 2018, 39(10): 11-21.
[20] VÖMEL S, FREILING F C. Correctness, atomicity, and integrity: defining criteria for forensically-sound memory acquisition[J]. Digital Investigation, 2012, 9(2): 125-137.
[21] HEO H S, SO B M, YANG I H, et al. Automated recovery of damaged audio files using deep neural networks[J]. Digital Investigation, 2019, 30: 117-126.
[22] 高元照, 李炳龙, 吴熙曦. 基于物理内存的注册表逆向重建取证分析算法[J]. 山东大学学报(理学版), 2016, 51(9): 127-136.
GAO Y Z, LI B L, WU X X. A forensic analysis algorithm of registry reverse reconstruction based on physical memory[J]. Journal of Shandong University (Natural Science), 2016, 51(9): 127-136.
[23] KHILOSIYA B, MAKADIYA K. Malware analysis and using memory forensic[J]. Multidisciplinary International Research Journal of Gujarat Technological University, 2020, 2(2):106-117.
[24] SCHUSTER A. Searching for processes and threads in Microsoft Windows memory dumps[J]. Digital Investigation, 2006, 3: 10-16.
[25] SALAVE P, WAKDIKAR A. Memory forensics: tools comparison[J]. International Journal of Science and Research, 2017, 6(6): 5-8.
[26] COHEN M. Scanning memory with Yara[J]. Digital Investigation, 2017, 20: 34-43.
[27] OKOLICA J, PETERSON G L. Windows operating systems agnostic memory analysis[J]. Digital Investigation, 2010, 7: 48-56.
[28] GS StatCounter. Desktop Windows version market share worldwide[R]. 2020.
[29] MARZIALE L, RICHARD G G III, ROUSSEV V III. Massive threading: using GPUs to increase the performance of digital forensics tools[J]. Digital Investigation, 2007, 4: 73-81.
[30] AL-SHARIF Z A, AL-KHALEE A Y, AL-SALEH M I, et al. Carving and clustering files in ram for memory forensics[J]. Far East Journal of Electronics and Communications, 2018, 18(5): 695-722.
[31] The Honeynet Project. Challenge 3-banking troubles[R]. 2010.

[作者简介]

李炳龙（1974− ），男，河南卫辉人，博士，信息工程大学副教授、硕士生导师，主要研究方向为数字调查与取证、网络入侵溯源追踪与取证、云计算取证、智能手机取证等。

周振宇（1976− ），男，河南太康人，博士，信息工程大学副教授，主要研究方向为信息安全。

张宇（1996− ），男，江苏连云港人，信息工程大学硕士生，主要研究方向为智能手机取证。

张和禹（1998− ），男，河南南阳人，信息工程大学硕士生，主要研究方向为内存取证。

常朝稳（1966− ），男，河南滑县人，博士，信息工程大学教授、博士生导师，主要研究方向为移动信息安全、物联网安全。