HZ BOOKS

PEARSON Addison Wesley

计算机取证

Forensic Discovery

（美） Dan Farmer
Wietse Venema 著

何泾沙 等译

机械工业出版社
China Machine Press

计算机取证

Forensic Discovery

（美） Dan Farmer
Wietse Venema 著
何泾沙 等译

机械工业出版社
China Machine Press

本书以重构过去事件为重点，目的是发现问题、分析问题、解决问题。本书分三部分，第一部分计算机对所涉及的基本概念进行介绍，包括以后章节中所用到的一些基本技术。第二部分计算机对文件系统、进程和操作系统的抽象进行了探讨。第三部分计算机主要对文件、进程和操作系统抽象之外的部分进行探讨。

本书面向那些想深入了解计算机系统的工作原理，以及想学习计算机入侵和系统分析技术的读者，适合计算机系统管理员、安全专家、开发人员等参考。

本书版权登记号：图字：01-2005-4696

图书在版编目（CIP）数据

计算机取证／（美）法默（Farmer, D），（美）温玛（Venema, W.）著；何泾沙等译. －北京：机械工业出版社，2007.5

书名原文：Forensic Discovery

ISBN 978-7-111-21241-6

Ⅰ.计… Ⅱ ①法… ②溫… ③何… Ⅲ 计算机犯罪－证据－调查－研究 Ⅳ. D 915.13

中国版本图书馆 CIP 数据核字（2007）第 043135 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：杨庆燕

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2007 年 5 月第 1 版第 1 次印刷

186mm × 240mm · 12.5 印张

定价：28.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

计算机科学以及信息技术的发展，尤其是近二十年来计算机网络技术的发展以及互联网应用的日益普及，已经使我们的日常工作和生活与数字信息紧密地联系在一起，信息以及信息技术无时无刻不在影响着我们。虽然我们从现代信息技术的应用中获取了诸多的益处，但我们同时也必须面临对信息进行恶意攻击和破坏给我们带来的种种不便以及可能造成的不可挽回的损失。因此，如何有效地使用与我们日常工作和生活密切相关的信息，不但通过直接使用而获益，而且通过对特定信息进行采集和分析去发现和阻止利用计算机进行犯罪以及对信息进行恶意攻击和破坏的企图和行为就成为我们必须面对的一个现实问题。这一问题充满了挑战，也是近年来一个极具活力的研究和开发领域。

计算机取证作为发现和阻止利用计算机进行犯罪的有效手段之一，近年来获得了高度重视。同时，多项研究和实践证明计算机取证是至今为数不多的发现和阻止对信息进行攻击和破坏的有效方法，获取的证据也具备了法律效力。然而，计算机取证所涉及的信息采集和分析具有较高的技术难度，同时也受到所涉及的计算机系统内部的实现和应用以及给用户提供的工具和服务的影响。因此，深入分析和了解计算机系统的内部体系结构，掌握取证的原理和常用方法以及编写和使用具体取证工具的方法对实现计算机取证起关键性作用。另一方面，由于目前得到广泛应用的计算机操作系统为 Windows、UNIX 和 Linux，深入研究和分析这些操作系统的内核对掌握计算机取证的原理和方法有事半功倍的效果。

本书从计算机操作系统入手，通过对文件系统和进程管理进行分析，着重阐述了系统内部对数据和信息进行处理的本质、数据在系统中持久存在的可能性以及如何利用以上特点对系统中的数据进行恢复以获取数字证据。作者在理论分析的基础上，结合多年积累的实践经验，提供了大量的计算机取证实例，实为理论与实际相结合的典范。本书面向希望系统学习计算机取证的基础理论和方法的人士，也适合对计算机取证的基本概念已经有了初步了解，但希望在此领域更加深入地开展具体工作的人士，

如编写计算机取证程序和工具、从事具体的数据分析和取证以及实际应用所获取的数字证据。因此，本书是一部深入浅出、信息量丰富的技术实用书籍，适合广大在企事业单位中从事信息安全方面工作的技术和管理人员进行阅读和系统学习。

本书由北京工业大学教授、博士生导师、软件学院常务副院长、北京市特聘教授何泾沙博士负责翻译和审校。李国瑞、付颖芳、吴旭和徐菲参与了部分章节的翻译和校对工作，全书最终由何泾沙进行统稿和审校。由于译者的水平有限，加上时间上的限制，本书的翻译难免存在不妥之处，敬请广大读者批评指正，译者在此深表谢意。

何泾沙

2007 年 5 月于北京

前　言

今天，计算机从接入因特网到被其他机器攻击只要几分钟时间，并且这种攻击只能算是背景噪声级别的无目的攻击。曾几何时，计算机可以年复一年地运行，从不被攻击。要了解因特网“背景辐射研究”的情况，请参考［CAIDA，2003］、［Cymru，2004］或［IMS，2004］。

在本书中，我们总结了过去几十年积累的“事后”入侵分析的经验，在此期间因特网呈爆炸性地增长，接入的主机已经从10万增长到了1亿［ISC，2004］。这种接入主机的增长，若不出所料的话，会导致更富戏剧性的现象——计算机和网络入侵更加频繁。随着网络特征和范围的改变，我们所面对的入侵的特征和范围也会随着改变。很高兴与读者共同分享这个学习机会。

然而，在这10年间，计算机处理信息的方式却几乎没有变化。事实上，我们可以肯定地宣称计算机系统在过去的35年里没有发生本质性的变化，包括整个因特网和Linux、Windows等其他现今使用的操作系统。尽管这些观察来源于如今的这些系统，我们仍乐观地期待其中的一些见识在未来的10年里将依然有效。

从本书能够学到什么

本书的前提是取证信息可以通过任何途径获得。在这个指导理念下，我们开发了搜集信息的工具，它能收集显式和隐式的信息。我们将详细分析真实的入侵，以及所用方法的局限性。

虽然本书列举了在特定系统环境中使用特定计算机取证工具的方法，但我们并不提供如何使用这些工具的细节说明，也不提供一步步核实的对照表。而是提供一些相关的背景资料，如信息如何维持，与过去事件有关的信息如何被恢复，以及信息的可信度如何受到进程有意或无意的影响。

在案例学习和例子中，我们不介绍传统计算机取证，而转向研究系统的动态特性。文件系统的易失性、持续性以及内存是贯穿本书的话题。我们的大部分例子来自Solaris、

FreeBSD 和 Linux 系统，微软的 Windows 也偶尔提到。我们强调的是这些系统共有的基本原理：寻找计算机系统的共性，而不是偶然的差异和表面特征。

我们通篇的主题以重构过去事件为重点，发现问题，分析问题，解决问题。这种方法可以帮助你弄明白事件为何会发生，但这基本上已经超出本书的范围。知道正在发生什么会使你对下次即将发生的恶性事件做好准备，即使当时了解的信息不足以预防未来问题的出现。必须预先提醒的是，我们并不会涉及入侵的检测和预防。我们将演示如何从跟踪一个入侵而导致发现另一个入侵，同时指出取证信息是怎样被系统保护机制和这些机制的错误操作所影响的。

目标读者

本书面向所有想要深入了解计算机系统是如何工作，以及所有想要涉猎计算机入侵和系统分析的技术员。系统管理员、应急响应人员、计算机安全专家以及取证分析员等都将受益于本书，当然包括所有关心计算机取证如何影响隐私信息在内的人员，也将受益于本书。

虽然我们尽量使本书适合非专业读者来阅读，然而本书并非面向计算机新手。我们假定读者已经深入了解 UNIX 或者 Windows 的文件系统、网络和进程的基本概念。

本书的结构

本书分三个部分：首先是基本原理，然后是进程、系统和文件的分析，最后以发现作为本书的结尾。读者可以不按照顺序阅读本书，但我们建议从第 1 章开始读起，因为它介绍了全书所涉及的所有基本概念。

在第一部分中，我们介绍通用的总体思路，以及后面章节所依赖的基本技术。

第 1 章介绍计算机结构的一般属性如何影响“事后”分析。阅读本章，可以帮读者预测后面遇到的所有的限制和意外。

第 2 章介绍时间线的基本概念，本章以基于主机和网络的信息为例，包含域名系统的信息，来介绍时间线的基本概念。我们观察了整整长达一年时间的入侵，并举例说明如何从不明显的地方获得时间信息。

在第二部分中，我们探索文件系统、进程和操作系统的抽象。这部分主要分析：如何理解计算机系统中找到的信息并且判断它的可信度。

第 3 章介绍文件系统的基本概念，以及后面要用到的取证工具和技术。

第4章通过详细检查一个受害机器的文件系统来描述一次入侵过程。我们查看现存文件和被删除的信息。如第2章那样，我们通过关联来决定不同观测结果的一致性。

第5章介绍用户进程和操作系统的执行环境。研究从直接更改系统工具的入侵，到几乎不可检测的内核模块的破坏入侵，以及对这些破坏的检测。

第6章介绍用于发现入侵后留下的进程或者程序文件目的的技术，以及防止恶意软件藏匿的对策和它们的限制。

在第三部分，我们着眼于文件、进程和操作系统抽象的束缚之外。和我们研究系统架构对信息破坏的影响一样，这部分主要集中在“发现”。

第7章说明那些大量被删除的文件可以完好地保存相当长的时间。我们发现它们大多都能在活跃的文件系统中存活2~4周。

第8章说明主存中信息的持久性，包括加密文件的解密内容。我们发现持久性具有很大不同，并把这些不同与操作系统架构属性关联起来。

附录提供了背景资料：附录A介绍Coroner's工具包及其相关软件；附录B给出目前从计算机系统中捕获计算机取证信息时的易失性顺序及其细节的相关知识。

本书的约定

我们使用$表示普通用户的shell命令提示符，#表示超级用户shell命令提示符。我们用大写字母开头的名字如Argus来表示一个系统而非独立的命令。本文中的UNIX包括Solaris、FreeBSD和Linux系统。在一些例子中，我们在命令提示符中包含了系统名称，如`solaris $`表明该示例只适用于Solaris系统。

如前面所提到的，本书中的许多示例均取自真实的入侵。为了保护隐私，我们对不属于我们自己的系统做了匿名处理。例如，使用私有网络地址代替真实网络地址，如10.0.0.1或192.168.0.1，并替换主机名和用户名等，甚至适当地替换了时间和时区。

网站

本书中的一些例子程序是为了发现和分析而编写的。我们无法在书中包含完整的代码清单，因为附加的细节致使偏离本书的目的。这些内容及其他程序的完整源代码可以在下面两个网站中找到：

http://www.fish2.com/forensics/

http://www.porcupine.org/forensics/

在这两个网站上，读者也可以找到一些免费的资料，如本书中没有包含的案例研究和其他的资源链接等。

致谢

我们非常感谢 Karen Gettman、Brian Kernighan 和 Addison-Wesley 出版社的其他人在我们撰写本书的几年间对我们的耐心支持。

尽管我们尽全力避免任何错误，但是没有我们的校阅团队，就没有这本书。我们特别感谢以下评审人员（按字母顺序）：Alephl、Brad Powell、Brian Carrier、Douglas Schales、Elizabeth Zwicky、Eoghan Casey、Fred Cohen、Gary McGraw、Muffy Barkocy、Rik Farrow、Steve Romig、Ben Pfaff。感谢 Ben Pfaff 和 Jim Chow 对于其中一章的贡献和 Dalya Sachs 对早期版本编辑所提供的帮助。感谢 Tsutomu Shimumura 鼓励我们去做超出我们能力的事情。Wietse 想感谢 FIRST 社团，感谢他们为本书提供一个可靠的公告板来形成和发展本书的很多思想和观念。本书与流行的写作方法不同，本书手稿先用 HTML 制作，然后用 vi 文本编辑器插入大量小的自定义脚本和标准的 UNIX 工具以 HTML 文档的形式完成的。

Dan Farmer

zen@ fish2. com

Wietse Venema

wietse@ porcupine. org

目　录

译者序
前言

第一部分　基本概念

第 1 章　计算机取证宗旨 …… 3
1.1　引言 …… 3
1.2　突显异常活动 …… 4
1.3　易失性顺序 …… 5
1.4　层与假象 …… 7
1.5　信息的可信度 …… 8
1.6　被删除信息的固化 …… 10
1.7　数字考古学与地质学 …… 11
第 2 章　时间机器 …… 15
2.1　引言 …… 15
2.2　故障的第一个特征 …… 15
2.3　MAC 时间介绍 …… 16
2.4　MAC 时间的局限性 …… 18
2.5　Argus：情况变得更为复杂 …… 19
2.6　淘金：在隐蔽的地方寻找时间信息 …… 23
2.7　DNS 和时间 …… 25
2.8　日志文件系统和 MAC 时间 …… 28
2.9　时间的缺陷 …… 31
2.10　结论 …… 32

第二部分　探讨系统抽象

第 3 章　文件系统基础 …… 35
3.1　引言 …… 35
3.2　文件系统的字母表 …… 35
3.3　UNIX 文件组织结构 …… 36
3.4　UNIX 文件名 …… 39
3.5　UNIX 路径名 …… 40
3.6　UNIX 文件类型 …… 41
3.6.1　普通文件 …… 41
3.6.2　目录 …… 41
3.6.3　符号链接 …… 42
3.6.4　IPC（进程间通信）端点 …… 42
3.6.5　设备文件 …… 42
3.7　首次揭密——文件系统内部情况 … 43
3.8　UNIX 文件系统布局 …… 49
3.9　揭开秘密——深入探索文件系统 … 49
3.10　模糊区——隐藏在文件系统接口之下的威胁 …… 51
3.11　结论 …… 52
第 4 章　文件系统分析 …… 53
4.1　引言 …… 53
4.2　初次接触 …… 53
4.3　准备分析被入侵的文件系统 …… 54
4.4　捕获被入侵的文件系统信息 …… 55
4.5　通过网络发送磁盘镜像 …… 56

4.6 在分析的机器上挂载磁盘镜像 …… 59
4.7 现存文件的 MAC 时间信息 …… 61
4.8 现存文件的详细分析 …… 63
4.9 掩盖现存文件分析 …… 65
4.10 插曲：当一个文件被删除时，将会发生什么？ …… 66
4.10.1 父目录项 …… 67
4.10.2 父目录属性 …… 68
4.10.3 索引节点块 …… 68
4.10.4 数据块 …… 68
4.11 被删除文件的 MAC 时间信息 …… 69
4.12 被删除文件的详细分析 …… 69
4.13 利用索引节点号发现异常文件 … 71
4.14 追踪一个被删除文件的原始位置 …… 72
4.15 通过被删除文件的索引节点号来追踪被删除的文件 …… 73
4.16 回到入侵的另外一个分支 …… 74
4.17 丧失无辜 …… 74
4.18 结论 …… 76
第 5 章 系统与破坏 …… 77
5.1 引言 …… 77
5.2 标准计算机系统结构 …… 78
5.3 UNIX 系统从启动到关闭的生命周期 …… 79
5.4 案例研究：系统启动的复杂性 …… 80
5.5 内核配置机制 …… 81
5.6 使用内核安全等级来保护计算机取证信息 …… 83
5.7 典型的进程和系统状态工具 …… 84
5.8 进程和系统状态工具是如何工作的 …… 87
5.9 进程和系统状态工具的局限性 …… 87
5.10 用 rootkit 软件进行破坏 …… 89
5.11 命令级破坏 …… 89
5.12 命令级的隐蔽和检测 …… 90
5.13 库级破坏 …… 93
5.14 内核级破坏 …… 94
5.15 内核 rootkit 的安装 …… 94
5.16 内核 rootkit 的操作 …… 95
5.17 内核 rootkit 的检测与隐藏 …… 97
5.18 结论 …… 101
第 6 章 恶意攻击软件分析基础 …… 103
6.1 引言 …… 103
6.2 动态程序分析的危险 …… 104
6.3 硬件虚拟机的程序限制 …… 104
6.4 软件虚拟机的程序限制 …… 105
6.5 软件虚拟机限制的危险性 …… 106
6.6 Jails 和 chroot( ) 的程序限制 …… 107
6.7 系统调用监控程序的动态分析 … 109
6.8 系统调用审查程序的限制 …… 111
6.9 系统调用哄骗程序的限制 …… 114
6.10 系统调用限制的危险 …… 116
6.11 库调用监控的动态分析 …… 117
6.12 库调用程序的限制 …… 117
6.13 库调用限制的危险 …… 120
6.14 机器指令级的动态分析 …… 120
6.15 静态分析与逆向工程 …… 120
6.16 小程序存在许多问题 …… 124
6.17 恶意攻击软件分析对策 …… 124
6.18 结论 …… 125

第三部分 超越抽象

第 7 章 被删除文件信息的持久性 … 129
7.1 引言 …… 129
7.2 被删除信息持久性举例 …… 130
7.3 测量被删除文件内容的持久性 … 131

7.4 测量被删除文件 MAC 时间的持久性 …… 132
7.5 被删除文件 MAC 时间的强力持久性 …… 133
7.6 被删除文件 MAC 时间信息的长期持久性 …… 136
7.7 用户活动对被删除文件的 MAC 时间信息的影响 …… 138
7.8 被删除文件信息的可信度 …… 139
7.9 为什么被删除文件信息能够保持不变 …… 140
7.10 结论 …… 142
第 8 章 超越进程 …… 145
8.1 引言 …… 145
8.2 虚拟内存的基础知识 …… 146
8.3 内存页的基础知识 …… 147
8.4 文件和内存页 …… 148
8.5 匿名内存页 …… 149
8.6 捕获内存 …… 149
8.7 savecore 命令 …… 150
8.7.1 内存设备文件：/dev/mem 和 /dev/kmem …… 151
8.7.2 交换分区 …… 152
8.7.3 其他存储单元 …… 153
8.8 静态分析：从文件中识别内存 … 154
8.9 在无密钥的情况下恢复加密文件的内容 …… 155
8.9.1 创建一个加密文件 …… 155
8.9.2 从主存中恢复加密文件 …… 155
8.10 文件系统块 VS. 内存分页技术 …… 156
8.11 识别内存中的文件 …… 158
8.12 动态分析：内存数据的持久性 … 159
8.13 内存中文件的持久性 …… 161
8.14 非文件或匿名数据的持久性 …… 163
8.15 交换分区的持久性 …… 164
8.16 引导进程内存的持久性 …… 164
8.17 内存数据的可信度和坚韧性 …… 165
8.18 结论 …… 167
附录 A Coroner's 工具包及其相关软件 …… 169
附录 B 数据收集和易失性顺序 …… 175
参考文献 …… 181

第一部分　基本概念

本部分的前两章，我们描述了本书的框架结构，并且介绍了在本书其他部分所运用的基本思想。用户行为对系统行为有什么影响？计算机体系结构和工具会带来什么影响？数据能持续多长时间？为什么持续这么长？为什么时间概念这么重要？

第 1 章是最容易理解的，也是最重要的一章。本章在一个相对高的层次上介绍了计算机取证的关键概念：易失性、层次和可信度，对于任何解释不是很清楚的地方，请先相信它们是对的，我们会在随后的章节里进行深入的阐述。

第 2 章以文件系统 MAC 时间、网络流量统计、甚至域名服务器为例，介绍时间线的概念。我们形成了对时间源及其存储位置的一种理解，并举例说明为什么我们强调主机中的数据，而不是网络上的数据，同时还列出了我们精心设计的第一个例子。

有经验的读者可以略过这一部分，而不用详细阅读它，但我们建议您还是至少粗略地浏览一下，因为这一部分提到的一些概念会出现在后面其他章节里面。

第1章

计算机取证宗旨

1.1 引言

关于寻找东西，这有几句话要说。当你去找某一个特定东西时，找到的可能性非常低。因为你是在找世上所有东西中的一个。而当你随便找任意一个时，找到的机会就非常的高。因为在世上所有东西中，你肯定能找到一些你要找的东西！

——Darryl Zeor, The Zero Effect

几年前，一个朋友哭着向我求助，有人入侵了她的 Solaris 系统，并且删除了大量的文件。为了帮助她，我们写出了文件恢复工具的最初版本，后来成为 Coroner's 工具包［Farmer, 2004］中的一部分。我的朋友只是想找回文件，而我们却想知道到底发生了什么。

我们没有期待恢复太多完整的东西。Solaris 像许多其他的 UNIX 系统一样，没有文件反删除功能。硬盘中被删的文件信息像个谜团一样放在那里，你得把零碎的部分重新放在一起。UNIX 的 FAQ 对此显得无能为力［FAQ, 2004］:

当你用“rm”命令删除一个文件时，无论出于什么目的，文件就这么消失了。一旦你“rm”一个文件，系统就会完全忘记分散在硬盘上的块中哪些是属于你的文件。更糟糕的是，这些曾属于你刚刚删除的文件的块将会在系统需要更多磁盘空间时最先被占用和写入。

当探讨那些被删除的文件时，我们发现一般的经验无能为力。首先，现代文件系统并不是随机地将文件内容分散在磁盘上。相反，在即使长达几年的频繁使用情况下，现代文件系统仍能成功地避免文件碎片的产生。其次，被删除的文件会完整地保留相当长的时间。更多有关被删除文件的持久性，可参看第7章。

接下来是我们所提倡的解决问题的一种典型方法：依靠以往的经验，听从他人的建议，并且使用现有的工具。当需要解决难题时，不要害怕把常识转变为神话，不要害怕创造自己的工具，不要害怕发展你自己的方法。否则，你可能会像那个因为路灯下比较明亮而只在路灯下寻找丢失的钥匙的人一样一无所获。本书的核心思想：如果你想学会解决问题，你就要有在任何地方找任何东西的准备，而且要在寻找前就准备好！

本章的余下部分对本书中出现的主要概念进行介绍。我们并不期望每位读者都有依次阅读每一章的耐心。你可以通过这一章了解到哪些是你最感兴趣的话题。

哦，恐怕我们忘了提到：后来，我的朋友的确找回了很多文件。

1.2 突显异常活动

每一个存储在系统中的比特到底会发生什么呢？大多数情况下什么也没发生。我们通过搜集不同UNIX服务器的数据来查看它们的文件被访问的频率。表1.1给出了一些因特网服务器文件的使用率和网络流量。

表1.1 一些因特网服务器文件的最近读写频率

	www. things. org	www. fish. com	news. earthlink. net
1年以上	76.6%	75.9%	10.9%
6个月~1年	7.6%	18.6%	7.2%
1~6个月	9.3%	0.7%	72.2%
1天~1个月	3.6%	3.1%	7.4%
24小时之内	2.9%	1.7%	2.3%

从表1.1可见，去年，两个典型的Web服务器上的绝大部分文件根本没使用过。即使在超负荷的Usenet新闻系统上（大部分根据用户需求定制），30天内使用过的文件也少于10%。不论它们是不是未使用的程序、配置文件、邮件文件、新闻以及数据等，总有很多文件会招“电子灰尘”。Windows PC和其他桌面系统也一样，我们发现90%的文件在过去一年内没有被访问过。

为什么会这样？即使是一台每秒能处理百万条指令的机器，在很短的时间内生成的新数据也足够装满 T 级容量的设备。当然，计算机相当忙碌，但多数操作都在重复访问相同的数据、程序和其他资源。当一个系统不断重复地访问同样的文件，它只不过在踩着自己的足迹而已。这也是为何异常活动的痕迹不但会显现出来，而且会显现很长一段时间！这是因为系统中大部分信息很少被动用！

几乎本书的所有章节都在讨论各种形式的数字痕迹。文件系统中痕迹的例子见第 2 章和第 4 章。第 8 章讨论内存中的数字痕迹。

1.3　易失性顺序

一个系统的取证分析首先是捕获数据，然后处理收集的信息。数据越正确完整，越能更好更全面地进行评价。原始数据保持其原始状态，并且所有分析只能对计算机数据的副本进行。这就有些类似于录下凶案现场，为保留证据而防止物理证据被破坏，允许其他人验证结论将伪造数据的可能性降到最低。

理想情况下，你需要整个系统及其所有数据的拷贝，但往往有各种障碍。当你收集数据时，系统的其他用户或者程序可能会改变系统状态或者销毁有价值的证据。情急之下，入侵者或极端分子可能设置电子地雷来破坏数据。仅仅执行一个程序，在加载并运行程序时，也会干扰计算机的状态。

正因如此，传统取证分析主要从静止系统中获取数据。按照惯例，采取关机并拷贝在这个事件中幸存下来的以下数据：程序日志、访问时间、文件内容等。随后分析这些拷贝的数据，以期减小对原数据的危害。这种方法方便了数据的捕获，同时为演示结果提供了不容否认的逻辑链。

我们一般提倡更好的理解而不是更高层次上的确定，这些因素可能在法庭上潜在地使一些方法论受到更多的置疑。相反，对于非确定因素——主要是数据收集方法，能够对其结论给予更多的了解和信任。这个过程需要统一收集数据的机制，同时要了解同一方法的各个方面。我们确信：为了得到可靠的结果，自动收集取证数据是有必要的。

当然，从一个运行中的系统中收集信息，需要谨慎有计划地进行。第一步，将计算机同其他用户和网络隔离。假如数据收集对某种类型的数据比其他数据的影响要小，那么收集数据的最好办法是按照数据预计的寿命进行收集。不同数据的生存期差别很大，从几纳秒（或更少）到几年。表 1.2 只不过列出了粗略的数据。