华章科技

ELSEVIER
爱思唯尔

世界级日志管理与分析专家亲笔撰写，完美诠释有效的日志分析策略以及最佳实践。

从日志的基本概念到系统运营，从传统的syslog到云计算和大数据环境下的新兴日志分析技术，全面解析日志管理和分析方面的各种实用技术及工具。

Logging and Log Management

The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management

日志管理与分析权威指南

（美）Anton A. Chuvakin　Kevin J. Schmidt　Christopher Phillips　著

姚军　简于涵　刘晖　等译

机械工业出版社
China Machine Press

作者向我们介绍了一种方法，它可以简化分析大规模日志数据的复杂过程，并且作者在书中对其推荐的日志管理和日志数据分析方法进行了详细的阐述。

—— **Reference and Research Book News**

三位作者通过本书向读者分享了他们在日志管理与分析方面的重要工作经验……对于那些希望从日志中寻找重要信息的读者，本书是一个巨大的资源，它展示了如何从大量日志数据中获得隐藏的信息，挖掘你需要的重要信息。

—— **RSA Conference**

对不同种类的日志进行高效管理与分析可能会面临很多难题，本书提供了大量最佳实践帮助你简化这个复杂的过程，包含系统、网络和安全日志中必须知道的信息，同时也详细地介绍了日志管理和日志分析方法，包括如何在系统和应用程序上创建实用的日志、日志搜索和日志审核。

本书主要内容

- 全面介绍日志管理，包括分析、可视化、报告等。
- 讲解了使用日志的各种信息——从系统运营到监管依从性。
- 包含日志在事故响应中的真实案例分析。
- 针对报告、日志分析系统选择、日志分析系统规划、日志数据规范化和关联等提供实用指南。

本书译自原版*Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management*并由Elsevier授权出版

ELSEVIER
爱思唯尔

投稿热线：(010) 88379604
客服热线：(010) 88378991 88361066
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn

ELSEVIER
指导：计算机/网络技术
ISBN 978-7-111-46918-6
9 787111 469186 >
定价：69.00元

华章程序员书库

Logging and Log Management

The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management

日志管理与分析权威指南

（美）Anton A. Chuvakin　Kevin J. Schmidt　Christopher Phillips　著

姚军　简于涵　刘晖　等译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

日志管理与分析权威指南 /（美）褚瓦金（Chuvakin, A. A.）等著；姚军等译 . —北京：机械工业出版社，2014.6
（华章程序员书库）
书名原文：Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management

ISBN 978-7-111-46918-6

I. 日… II. ①褚… ②姚… III. 计算机网络管理－指南 IV. TP393.07-62

中国版本图书馆 CIP 数据核字（2014）第 116145 号

本书版权登记号：图字：01-2013-6487

Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management

Anton A. Chuvakin, Kevin J. Schmidt, Christopher Phillips

ISBN: 978-1-59749-635-3

日志管理与分析权威指南

[美] Anton A. Chuvakin 等著

出版发行：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：吴 怡　　责任校对：董纪丽
印　　刷：三河市宏图印务有限公司　　版　　次：2014 年 6 月第 1 版第 1 次印刷
开　　本：186mm×240mm 1/16　　印　　张：21
书　　号：ISBN 978-7-111-46918-6　　定　　价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066　　投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259　　读者信箱：hzjsj@hzbook.com

The Translator's Words 译者序

对于广大 IT 工作者（尤其是负责运维工作的人）来说，日志是一个熟悉的名词，机房中的各种系统、防火墙、交换机、路由器，都在不断地产生日志，而在我们的规章制度上，还有着手工记录或者由某些代理程序采集的运维日志。但是，就是这个天天伴随我们的概念，我们真的熟悉吗？我们是否天天都会去查看重要系统的日志，在系统出现问题或者问题隐患的时候，我们是否能从日志的分析中看出端倪？当系统遭受严重的攻击和破坏时，我们的日志记录系统是否能够幸免于难，帮助我们一步步走出泥淖？

IT 安全业界的无数实践告诉我们，健全的日志记录和分析系统是系统正常运营与优化以及安全事故响应的基础，虽然安全系统供应商为我们提供了五花八门的解决方案，但是最终的基础仍是具有充足性、可用性、安全性的日志记录系统。作为运维人员，以及为企业开发应用程序（日志生产者）的开发人员，能不能回答上述的问题，正是对我们实际工作能力的一个考验。

在实际工作中，我们可以看到，许多单位内部对日志并没有充分的认识，安全工作更多地投入设备，比如防火墙、IDS、IPS、防病毒软件等，被动地希望这些系统帮助我们完成一切工作，但是俗话说得好："道高一尺，魔高一丈"，以特征码和预定义规则为基础的上述设备，在防护方面永远落在攻击者的后面，防微杜渐才是真正的出路。作为一名合格的运维人员，了解日志的概念，了解日志的配置和分析方法，是发现威胁、抵御攻击的重要技能，有了这方面的深刻认识，各种自动化安全解决方案才能真正地发挥效能。

多年以来，许多安全书籍都会或多或少地提及日志的作用，但是我们都苦于没有一本全面介绍日志的书籍，这也给人们在日常工作中开发有效的日志记录、分析、响应系统，以及运营中相关的规程带来了困难。

本书是由三位业界资深的安全专家编撰，从日志的基本概念开始，由浅入深地讲述了整个日志生命期的详细过程，书中以大量实例介绍了许多日志方面的最佳实践，从传统的

syslog，一直到正在不断兴起的云计算和大数据，涵盖了这一重要领域中的各项法则，并结合业界中最为常见的监管规章。不仅从详细的技术及工具上，而且从整个运营规程、策略上形成了完整的系统，从而突破了行业和具体软硬件配置的限制，不管读者身处何种规模、何种软硬件配置，均能从本书介绍的概念和思路中获益，并通过自己的努力，形成基于标准、适合自身特点的日志运营架构。

本书涵盖的内容极广，给翻译工作带来了很大的挑战，同时也让译者受益匪浅，我们衷心地希望书中的内容能给广大读者带来帮助，由于译者水平所限，错误在所难免，期待读者朋友们的指正。本书的翻译工作主要由姚军完成，简于涵、刘晖、徐锋、陈绍继、郑端、吴兰陟、施游、林起浪、陈志勇、刘建林、白龙、林耀成、陈霞、方翊、宁懿等人也为本书的翻译工作做出了贡献，在此衷心感谢华章公司的编辑们对翻译工作提出的中肯意见，以及其他工作人员对本书付出的艰辛努力。

About the Author 作者简介

Anton A. Chuvakin 博士是日志管理、SIEM 和 PCI DSS 依从性领域公认的安全专家，他参与撰写了《Security Warrior》(ISBN: 978-0-596-00545-0) 和《Know Your Enemy: Learning About Security Threats》第 2 版（ISBN: 978-0-321-16646-3）、《Information Security Management Handbook》 第 6 版（ISBN: 978-0-8493-7495-1）、《Hacker's Challenge 3:20 Brand-New Forensic Scenarios & Solutions》（ISBN: 978-0-072-26304-6）、《OSSEC Host-Based Intrusion Detection Guide》(Syngress，ISBN: 978-1-59749-240-9）等书籍。

Anton 已经发表了数十篇有关日志管理、关联分析、数据分析、PCI DSS、安全管理等安全主题的文章。他的博客 www.securitywarrior.org 是该领域中最受欢迎的博客之一。此外，Anton 在全球的许多安全会议上发表演讲，包括美国、英国、新加坡、西班牙、俄罗斯等地。他参与新兴的安全标准的制定，并且担任多家安全领域创业公司的顾问。

目前，他运营自己的顾问公司 Security Warrior。在此之前，他曾经是 Qualys 的 PCI 依从性解决方案主管和 LogLogic 的首席日志管理者，任务是为全世界提供关于安全、标准化和运营日志的重要性的培训。在 LogLogic 之前，他曾经受雇于一家安全供应商，担任战略产品管理职务。Anton 拥有 Stony Brook 大学的博士学位。

Kevin J. Schmidt 是 Dell SecureWorks 公司的高级经理，这家业界领先的安全托管服务提供商（MSSP）是 Dell 的下属公司。他负责公司 SIEM 平台主要部分的设计和开发，包括数据获取、关联分析和日志数据分析。就职于 SecureWorks 之前，Kevin 为 Reflex Security 工作，致力于 IPS 引擎和反病毒软件。在此之前，他是 GuradedNet 公司的首席开发人员和架构师，该公司构建了行业最早的 SIEM 平台之一。他还是美国海军预备队（USNR）的军官。Kevin 在软件开发和设计领域有 19 年的经验，其中 11 年从事网络安全领域的研发工作。他持有计算机科学学士学位。

Christopher Phillips 是 Dell SecureWorks 的经理和高级软件开发人员，负责公司 Threat

Intelligence 服务平台的设计和开发。他还负责一个团队，致力于集成来自许多第三方提供商的日志和事件信息，帮助客户通过 Dell SecureWorks 系统和安全专业人士分析信息。在就职于 Dell SecureWorks 之前，他为 McKesson 和 Allscripts 工作，帮助客户进行 HIPAA 标准化、安全性和保健系统集成方面的工作。他在软件开发和设计领域有 18 年以上的经验，持有计算机科学学士学位和 MBA 学位。

技术编辑简介

Patricia Moulder（CISSP、CISM、NSA-IAM）是一位高级安全主题专家和顾问。她持有东卡罗莱纳大学科学硕士学位。她在网络安全评估、Web 应用审计、商用及美国政府客户无线网络技术方面有超过 19 年的经验。她在辛克莱尔社区学院担任网络安全助理教授 5 年之久，她在 SDLC 应用安全审计和数据隐私标准化方面也有大量跨平台经验。

Preface 序　言

从我第一次遇到 syslog 算起已经有 25 年了。当时我是由围绕一台 Sun-3 新组建的 Sun-2 小型群集系统的管理员，试图通过电话和一位朋友一起调试 UUCP 连接，这位朋友告诉我“检查日志”。观察 syslog 有时令我昏昏欲睡，但我发现，这是查看计算机实际上“做了什么”的最好办法。分屏系统可以在屏幕的上角打开一个窗口运行“ tail–f/usr/spool/messages”命令，这展示了其价值；我可以观察电子邮件的工作、USENET 新闻流、进程启动和停止，我可以看到自己的计算机实际在做什么！直到几年之后我第一次遇到安全事故，我才发现日志对于观察过去和当前的情况都很有用。之后，系统管理员在屏幕的一角观察其日志的时代已经一去不复返，现在，这一切已经很模糊了。

为什么我们当中的有些人痴迷于日志，而其他人则对其漠不关心？我认为这是因为，系统管理员有可能认为“日志记录了某些事情”因而“计算机做了某些事情”以及“一切正常，所以我很高兴”。我的第一个日志分析算法很简单：

```
If the syslog stops, the Pyramid's I/O processor is wedged again
```

如果日志工具能够一次又一次地帮助你摆脱困境，你就会不断地使用它。25 年来，我从 syslog 开始，用日志工具完成了如下工作：

- 搜寻对某个主要电子商务零售商发动“购物车穷举攻击”的痕迹。
- 找出 1000 多个系统中恶意软件自动投放的位置。
- 将我暑假中的一些时光花费在分析一家超级计算机中心 10 年的日志，偶然地发现只通过日志的体积就能够检测主要的 Linux 发行版本。
- 构建一个网站数据复制系统，使用 syslog 作为原子事务记录。
- 确认谁向 President Clinton @whitehouse.gov 发送了一封危险电子邮件。
- 计算我的软件工程师在 Diablo Ⅱ游戏上花了多少时间。
- 解析和重放 6 个月的日志中的事务，重新构造一个损坏的数据库。

除此之外，我还用 syslog 做许多“日常工作”，例如确保系统一切正常，寻找不寻常的活动，以及找出故障点及其原因。和计算机中的其他系统不同，我们可以用系统日志对过

去的工作进行限制性查看，限制来自我们收集的信息，以及保存的时长，这是一个有用的视图。因此我总是说："如果你看到一个系统的日志是关闭的，就说明系统管理员对其工作毫不在乎。"。我不知道曾经有多少次听到在事故响应中，关键的系统"因为性能的原因"关闭了日志。对于我这样痴迷于日志的人来说，那样的做法完全没有经过深思：性能问题很容易用更快的处理器或者一个闪存盘来解决，而没有了日志，你就是在"盲飞"。

奇怪的是，关于系统日志的好书并不多。你可能以为，有许多的书籍能够告诉你有用的日志是什么样的，但是实际上没有，这可能是因为该主题有些枯燥，而且这个主题太过广泛，需要覆盖从获取数据到如何处理所获取数据的所有细节。而且还有一个大问题：如何处理日志本身没有简单的方法。"在星期天需要对 syslog 做的前 10 种处理"这样的简单规则是没有的，因为每个人的日志都不相同，需求也各不相同。写一本从本质上说明"重点，查看你的日志并且思考你所看到的内容"的书籍是很困难的。安全人员想要看的是入侵的企图；系统管理员看到的是表示系统正常运行的特征；CIO 将会查看使用情况指标和业务证据；审计人员将会查看可以选择的复选框，凡此种种。窍门就是向所有这些人解释：系统日志是一种集地板蜡、甜点配料和足底按摩器为一体的全能产品，哦，顺便提一句：它只适用于自己动手解决问题的人。

日志需要自己动手这种特性可能是日志难以令人喜欢的原因。而这本书正是你们所要寻找的日志"菜谱"，但是没有简单的"通用"技巧。你必须思考每一种思路，然后对其进行改编，应用到你的特殊情况中。当我教授系统日志分析课程（很久以前的事了！）时，我总是发现班上的有些学生很不愉快地离开了：他们期待在离开时带走"真正能揭示一切的饼状图"或者"具备强大查找功能的日志分析 Perl 脚本"。与此相反，我讲的是通过数据思考分析和分类的框架。我曾经在课程的先决条件中要求"必须了解某种编程方法"，还记得有一位学生最喜欢的日志分析编程语言是 MATLAB。不管你得到什么数据，你所要寻找的就是工作的最佳工具。

在序言中加入我自己的建议可能不太合适，但是无论如何我还是要这么做。对日志进行有效操作的最佳途径是召集 3 ~ 4 位能干的人一起开会，买一些披萨和啤酒，花上两个小时审视你的日志。将日志的内容投影到大屏幕上，让每个人都能看见，并且来回滚动，查看你所得到的结果。然后，当你觉得厌倦的时候开始问自己：你从日志中得到了哪些想要知道的信息，你想要生成哪些方面的摘要，哪些内容能够组成系统工作完成的有用指标，哪些内容可能指出关键的错误。本书对于如何做到这些提供了更多有用的细节，但是相信我，买些披萨和啤酒是没错的。

我可能已经说了太多你已经知道的事情，所以该到此为止了。现在，翻过这一页，开始阅读本书吧！

Marcus J.Ranum

Tenable Network Security 公司 CSO

Preface 前　言

欢迎阅读本书。本书的目标是向信息技术（IT）专业人士提供理解和处理日志数据的入门知识。各种形式的日志数据是由许多类型的系统生成的。如何处理和分析日志数据是长期存在的一个问题。本书介绍能够帮助你分析日志数据和寻找恶意活动的技术和工具。

过去，系统管理员审阅日志文件，寻找磁盘错误或者内核问题。现在的系统管理员往往还要兼任安全管理员。更好地理解如何处理安全日志数据的需求从未像今天那么重要。安全性分析人员是IT专家组中负责跟踪日志分析技术的人。许多经验丰富的人曾经在“压力测试”的模式下进行学习。本书的目标是提供帮助你快速理解各种概念的材料，提取许多人曾经花费多年学习的知识。

我们先来谈谈最近的一个重要问题：法规依从性。随着安能公司和其他企业的垮台，法规依从性现在成为了许多企业的核心主题，焦点现在集中在政策和规程。作为IT工程师，你能说明，Bob在被解雇之后无法访问自己的企业电子邮件账户吗？这些事情需要公司提出规程。系统和网络日志的框架正在以这样那样的方式变化。

目标读者

本书面向任何对学习日志记录和日志管理感兴趣的人。下面介绍一些应该阅读本书的人士。

系统管理员：承担企业日志数据监控任务。

初级安全工程师：网络安全的新手，希望学习日志分析技术。

应用程序开发人员：对从头构建一个日志分析系统感兴趣的人，本书提供了这方面的示例代码。但是，全书为日志分析的重要性提供了很好的背景资料，这些领域也不应该忽略。

管理人员：想深入理解日志数据收集、存储、分析和法规依从性等主题的人。如前所述，这些问题在企业中越来越多地出现，并将持续地成为IT专业人士需要更多关注的领域。

所需基础知识

我们假定读者对网络、操作系统和网络安全等概念有基本的了解。但是，理解本书中的素材并不要求读者是计算机科学家或者网络高手。需要背景信息的主题提供了必要的细节。大部分代码示例采用 Perl 和 Java 编程语言。理解或者使用这些代码示例并不要求读者是 Java 高手，所以我鼓励每个人都仔细地阅读这些例子。

本书的组织

本书的每一章内容都建立在前一章的基础上。尽管如此，许多章节可以独立阅读。本书共有 22 章。

第 1 章：木材、树木、森林

该章提供了日志系统的背景信息。如果你熟悉了 syslog、SNMP、安全日志、日志数据收集、存储等概念，就可以跳过本章。

第 2 章：日志是什么

该章描述日志消息，包括日志重要性的讨论。

第 3 章：日志数据来源

该章描述 syslog 协议、SNMP 和 Windows 事件日志。此外，介绍了日志数据源的分类。

第 4 章：日志存储技术

如果你想要学习有关日志保留、存储格式和在关系数据库管理系统（RDBM）中存储日志的知识，那么请详读该章。我们甚至提供了使用 Hadoop 进行这方面工作的例子。

第 5 章：syslog-ng 案例研究

该章深入介绍了如何在实际环境中部署 syslog-ng 进行日志收集，还讨论了 syslog-ng 的一些较为高级的功能。

第 6 章：隐蔽日志

如果你需要以隐蔽的方式使用日志，该章提供了完成这一任务的许多细节。

第 7 章：分析日志的目标、规划和准备

在你开始分析日志数据之前，首先需要设定目标、规划并为任务做准备。该章介绍的主题包括寻找过去的问题、未来的问题和从未见过的情况。

第 8 章：简单分析技术

在讨论高级分析技术之前，需要介绍基础知识，这包括人工日志分析及其工具。除此之外，我们讨论一个可以使 Windows 事件日志的阅读更为简单的高级工具。当然，我们还讨论了响应日志分析结果的过程。

第 9 章：过滤、规范化和关联

该章是激动人心的一章，介绍了相关的技术和工具，能够帮助你执行关联分析，找出简单的人工日志分析可能忽视的问题。该章介绍的主题包括过滤、规范化、分类学、关联和一些常见的搜索模式。该章中有两节是为对构建自己的关联分析引擎感兴趣的开发人员而写的。该章还介绍了 Jess 和 Esper，说明如何构建基于规则和基于流的引擎。

第 10 章：统计分析

该章讨论如何使用统计执行分析，介绍了频率统计、基线、阈值和异常检测。我们甚至提供了使用机器学习进行分析的方法。

第 11 章：日志数据挖掘

该章专门介绍日志挖掘或者日志知识发现——不同类型的日志分析，它不依赖于对所要寻找的内容的了解。这打破了日志搜索中对字符串列表或者模式的依赖，将日志分析的“高级艺术”提高到一个新的水平。

第 12 章：报告和总结

该章关注作为日志分析手段的报告，我们特别重视对日志数据最佳报告的定义。

第 13 章：日志数据可视化

日志数据的可视化往往很有用。我们的意思不是查看警报、电子邮件或者特定日志分析系统的输出。我们更感兴趣的是讨论在有向图和其他可视化工具中查看日志数据。

第 14 章：日志法则和日志错误

该章介绍各类组织在日志上常见的错误（以及正在犯的错误），还介绍一些组织处理日志中常见的规则和相关性，“法则”的叫法可能过于严格了。

第 15 章：日志分析和收集工具

该章提供了用于日志数据分析和收集的开源和商用工具的评论，这些评论可为读者在选择日常数据管理工具时提供一些参考信息，包括使用这些工具进行日志分析的例子，并且包含了使用这些工具审核常见日志任务和场景的实例。该章将帮助读者评估可用的工具集，以便确定自己组织中分析日志的合适工具。

第 16 章：日志管理规程

该章对日志管理中的日志审核、响应和升级做了介绍。使用支付卡行业（Payment Card Industry，PCI）数据安全性标准（Data Security Standard，DSS）的例子是该章的主题。思路是阐述如何在现实中应用概念。这意味着，示例针对 PCI 标准，但是它们很容易经过改编，扩展到任何环境中。本质上，该章开发了你可以使用的一组步骤和规程。额外的一个好处是对日志管理中解读和应用标准的深入了解。

第 17 章：对日志系统的攻击

该章介绍了对日志记录、日志分析系统的攻击，甚至日志分析人员对日志安全、操作和依从性的破坏。

第 18 章：供程序员使用的日志

该章对于各种程序员都是实用的。这包括系统管理员、Perl 程序员、C/C++ 程序员、Java 程序员及其他编程人员。本质上，任何编写脚本、程序或者软件系统的人都将从该章的内容中获益。人们常说，糟糕的日志消息是糟糕的程序员造成的。虽然这不完全对，但该章的目的就是通过向程序员和其他人提供如何生成更好的日志消息的指导和概念，进而改变这一看法。最终，这些方法将对调试、信息收集和解析能力带来帮助，并且增进软件生成的日志消息的整体实用性。

第 19 章：日志和依从性

该章介绍日志记录和法规及政策的依从性，该章对于任何苦于应付法规依从性的人来说都是有价值的。

第 20 章：规划自己的日志分析系统

该章为日志分析系统的部署规划提供实用的指南。该章的目的不是提供安装特定日志分析系统的详细蓝图，而是提供素材，使你能够将概念应用到任何自己所处的日志分析部署中。该章将告诉你在这一过程中所要询问的问题和考虑的项目。

第 21 章：云日志

云计算是现在的热门话题，并将越来越热门。我们目睹了传统的紧缩型套装产品从公司所属的数据中心移植到云中（这一切正在发生），IT 管理员在硬件、交换机、机架、软件上花费较少的资本性支出（CAPEX），控制日志数据收集、集中和存储甚至安全信息及事件管理（SIEM）的机会将会大大减少。该章介绍云计算和云日志，还介绍与云环境相关的法规和安全问题、云中的大数据、云中的 SIEM、优缺点以及几个关键云日志提供商的详细清单。

第 22 章：日志标准和未来的趋势

该章提供了日志标准的未来和日志记录及日志分析未来发展的专业意见。

致谢

Dr.Anton A.Chuvakin

首先，也是最重要的：感谢我的妻子 Olga，她是我所有作品永恒的灵感源泉，感谢她提供的宝贵的项目管理建议，感谢她容忍（是的，几乎总是在容忍……）我将本可以共度的夜晚用在本书上。

接下来，我特别感谢 Marcus Ranum 为本书所作的序。

最后，感谢 Syngress/Elsevier 的全体人员，感谢他们对我们未能按照承诺的时间交稿的宽容。

Kevin J.Schmidt

首先，我要感谢美丽的妻子 Michelle。她给我的鼓励和支持帮助我完成了本书。当然，感谢我的雇主 Dell，为我完成这一项目提供了支持。接下来必须感谢提供宝贵意见的同事们：Rob Scudiere、Wayne Haber、Raj Bandyopadhyay、Emily Friese、Rafael Guerrero-Platero 和 Maro Arguedas。来自 BalaBit IT Security 的 Robert Fekete 在 syslog-ng 的有关章节上提供了杰出的意见。Ernest Friedman-Hill 为第 9 章中关于 Jess 的小节提供了宝贵的建议。我过去的同事 Jimmy Alderson 慷慨地为第 13 章提供了代码示例。最后，我要感谢合著者 Anton 和 Chris，感谢他们为这本出色的书籍提供了杰出的内容。

Christopher Phillips

我要感谢美丽的妻子 Inna 和可爱的孩子们，Jacqueline 和 Josephine。她们的亲切、幽默和爱，在我编写本书和经历生活中所有努力和冒险的时候提供了灵感和支持。我还要感谢我的父亲一如既往地支持和鼓励我从事工程和科技。感谢 Rob Scudiere、Wayne Haber 和我的雇主 Dell 提供的宝贵意见。我还要特别感谢合著者 Kevin 提供机会，让我成为这本出色书籍的作者之一。Kevin 在 Dell SecureWorks 和我共事的许多年中提供了很好的指导和鼓励，帮助我在职业生涯中变得更为专业。他的领导能力和安全知识是我、我们的客户和每天在一起的许多人灵感的源泉。

目　录 *Contents*

第 1 章 Chapter 1

木材、树木、森林

本章要点

- 日志数据基础
- 接下来的事情
- 日志被低估了
- 日志很有用
- 人、过程、技术
- 安全信息和事件管理 (SIEM)
- 案例研究

1.1 概述

本书是关于如何处理系统日志的。更确切地说，是关于如何从你的日志中获取到各种有用的信息。经常被低估的日志其实是计算机系统资源管理（打印机、磁盘系统、电源备份系统、操作系统等）、用户和应用程序管理（登录注销、应用程序访问等）和安全的一种非常有用的信息来源。需要说明的是，信息经常可以不同的方式来进行分类。用户的登录和注销信息与用户管理和安全都有关系。下面用一些例子来展示日志信息是多么有用。

各种磁盘存储产品在硬件出错时都会记录日志，经常关注这些信息可以把小问题在变成真正的噩梦之前解决掉。

作为第二个例子，让我们来简单思考一下，用户管理和安全日志如何结合起来为用户

活动提供线索。当一个用户登录到 Windows 环境中时，这个行为已经作为登录历史被记录到某处。我们称之为用户管理日志数据。无论何时用户访问网络的任何部分，防火墙很有可能已经起作用了。防火墙，同时也记录着来源（例如个人工作站）是否有权限向一个特定的网络组件发送网络包，我们称之为安全日志数据。现在，让我们假设你的公司正在开发一个新产品，你想要知道谁尝试过访问你的开发服务器。当然，你可以采用防火墙访问控制列表（ACL）来管理，不过你想更进一步，用户的登录数据可以和防火墙记录对应起来，显示出尝试访问服务器的用户。如果这并不是在正常业务时间发生的，你就有理由来和这些员工谈谈以便更好地了解他们的意图。更深入一点，这个例子传达出一个重点，如果你可以访问正确的信息，你就有能力做一些精确的事情。

但是，获取这些信息需要花费一些时间和精力。第一眼（可能第二眼也是如此）看起来会是一项非常艰巨的任务，庞大的日志数据单独看起来也会令人望而生畏。不过我们能帮助你来“抵抗”它。我们会展示给你一个处理日志的完整策略，也会展示一些不同的日志类型和格式。使用不同类型和格式日志有着双重的意义。首先，它会让你习惯于查看日志消息和数据，让你更加熟悉它们。其次，它会帮助你建立一种心态：理解基本的日志格式会使你更加容易地识别和处理在你的环境里新的或者以前没见过的日志数据。不同的供应商会生成不同格式的日志消息，这是一个残酷的事实，但是最终，真正的关键是你如何处理和管理日志数据，越快理解并将它集成到你的综合日志系统，你就能越早地从中获利。

本章的余下部分是本书其他部分使用到的概念的基础。我们将要探究围绕着日志数据、人、过程和科技的思想，穿插进一些真实的例子使你发现日志数据的真正价值。

1.2 日志数据基础

到目前为止，我们一直没有对所讨论的日志记录和日志数据提供一个确切的说明。现在，让我们来详细地介绍围绕着日志记录和日志数据的基础知识。

1.2.1 什么是日志数据

简单地说，日志数据的核心就是日志消息或者日志。日志消息就是计算机系统、设备、软件等在某种刺激下反应生成的东西。确切的刺激在很大程度上取决于日志消息的来源。例如，Unix 操作系统会记录用户登录和注销的消息，防火墙将记录 ACL 通过和拒绝的消息，磁盘存储系统在故障发生或者在某些系统认为将会发生故障的情况下生成日志信息。

日志数据（log data）就是一条日志消息的内在含义。换句话说，日志数据就是一条日志消息里用来告诉你为什么生成日志消息的信息。例如，Web 服务器一般会在有人访问 Web 页面请求资源（图片、文件等等）的时候记录日志。如果用户访问的页面需要通过认证，日志消息将会包含用户名。这就是日志数据的一个例子：可以使用用户名来判断谁访问过一个资源。

日志（log）这个术语实际上指的是用于展示某些事件全貌的日志消息的集合。

日志消息可以分成下面的几种通用类型：

- 信息：这种类型的消息被设计成告诉用户和管理员一些没有风险的事情发生了。例如，Cisco IOS 将在系统重启的时候生成消息。不过，一定要当心。例如，如果重启发生在非正常维护时间或是业务时间，就有发出警报的理由。本书后面的章节将会提供给你检测此类现象发生的技能和技术。
- 调试：软件系统在应用程序代码运行时生成调试信息，是为了给软件开发人员提供故障检测和定位问题的帮助。
- 警告：警告消息是在系统需要或者丢失东西，而又不影响操作系统的情况下生成的。例如，如果一个程序没有获得正确数量的命令行参数，但是它也能在没有这些参数的情况下运行，这种情况下程序记录日志可能只是为了警告用户或者操作人员。
- 错误：错误日志消息是用来传达在计算机系统中出现的各种级别的错误。例如，操作系统在无法同步缓冲区到磁盘的时候会生成错误信息。不幸的是，许多错误消息只能给出为什么出错的起点，要寻找出导致错误发生的根本原因还需要进一步的调查。本书的第 7、8、9、10、11、12、13、15 和 16 章将为你提供处理这些情况的方法。
- 警报：警报表明发生了一些有趣的事情。一般情况下，警报是属于安全设备和安全相关系统领域的，但这并不是硬性的规定。在计算机网络中可能会运行一个入侵预防系统（IPS），检查所有入站的流量。它将根据数据包的内容判断是否允许其进行网络连接。如果 IPS 检测到了一个恶意连接，可能会采取任何预先配置的行动。IPS 会记录下检测结果以及所采取的行动。

接下来，我们要简短的讨论一下日志数据是如何传输和收集的，然后我们会谈到日志消息的组成。

1.2.2　日志数据是如何传输和收集的

日志数据的传输和收集在概念上非常简单。计算机或者其他设备都实现了日志记录子系统，能够在确定有必要的时候生成日志消息，具体的确定方式取决于设备。例如，你可以选择对设备进行配置，设备也可能本身进行了硬编码，生成一系列预设消息。另一方面，你必须有一个用来接收和收集日志消息的地方。这个地方一般被称为日志主机（loghost）。日志主机是一个计算机系统，一般来说可能是 Unix 系统或者 Windows 服务器系统，它是集中收集日志消息的地方。使用集中日志收集器的优点如下：

- 可以集中存储从多个地点得到的日志消息。
- 可以在上面备份你的日志。
- 可以在上面进行日志数据的分析。

这固然是好事，但是，最初的日志消息是如何传输的呢？最常见的方法是通过 syslog 协议。syslog 协议是日志消息交换的一种标准。它常见于 Unix 系统中，也存在于

Windows 和其他非基于 Unix 的平台上。syslog 基本上都实现了客户端和服务器端组件，两者之间通过用户数据报协议（UDP）通信，但是为了可靠传输，很多开源和商业 syslog 实现同样也支持传输控制协议（TCP）。客户端部分是生成和发送日志消息的真实设备或者计算机系统。服务器端通常会在一个日志收集服务器上。它的主要工作就是采集基于 syslog 的日志消息并将其存储在一个本地磁盘上，在那里可以分析日志，备份以及存储以供长期使用。

syslog 并不是传输和收集日志数据的唯一机制。例如，微软为 Windows 开发了自己的日志记录系统，称做 Window 事件日志（Windows Event Log）。用户登录注销、应用程序消息等都以专有的格式存储。有开源和商业的应用程序用来将 Windows 事件日志转换成 syslog 的格式，以发送给 syslog 服务器。我们将会在第 3 和第 15 章对 Windows 事件日志进行更详细的讨论。

简单网络管理协议（SNMP）是一种用来管理网络设备的基于标准的协议。此协议基于两个概念：陷阱和轮询。陷阱就是当某些事情发生的时候，设备或者计算机系统发出的日志消息的一种形式。陷阱发送到管理站——这类似于日志主机。管理站用来管理基于 SNMP 的系统。轮询是管理站使用 SNMP 来查询设备预设变量（例如接口统计数据、接口的进出字节数等等）的手段。SNMP 和 syslog 主要的区别是 SNMP 应该是根据数据格式来组织的，但在实践中并不总能找到。如果您想要了解关于 SNMP 的更多内容，请参阅《Essential SNMP》(Mauro & Schmidt，2005)。

数据库已经变成了应用程序存储日志消息的简便途径。应用程序可以将它的日志消息写进数据库模式，而不是生成一条 syslog 消息。在某些情况下，syslog 服务器本身也可以直接写入关系型数据库，特别是在结构化存储、分析和报告日志消息的情况下有着极大的优势。

最后，也有一些专有的日志记录格式。第三方设备和应用程序实现了用于生成和检索日志消息的专有机制。在这个领域，供应商可能以 C 或者 Java 类库的形式给你提供应用编程接口（API），或者由你自行实现协议。可将 Windows 事件日志看作一种专有格式，但时常人们将其看作非官方日志记录标准，类似 syslog，因为它相当流行。

我们在本节中讨论了以下一些较为常见的协议：

- syslog：基于 UDP 的客户端 / 服务器协议。这是最常见和普遍的日志记录机制。
- SNMP：SNMP 最初是为了管理网络中的设备而创造的。然而多年来许多非网络系统已采用 SNMP 作为发出日志消息和其他状态类型数据的方式。
- Windows 事件日志：微软的专有日志记录格式。
- 数据库：以结构化的方式来存储和检索日志消息。
- 常用的专有协议：
  - LEA：日志提取 API（Log Extraction API，LEA）是 Checkpoint 用于从它的防火墙和安全产品线收集日志的 API。

- SDEE：安全设备事件交换（Security Device Event Exchange，SDEE）是思科用于从它的 IPS（入侵预防系统）设备产品线收集日志消息的基于可扩展标记语言（XML）的协议。
- E-Streamer：E-Streamer 是 Sourcefire 公司为其 IPS 开发的专有协议。

提示　注意支持的日志记录方法

有些设备支持多种日志记录方法。例如，IPS 设备可能支持将日志记录到 syslog 服务器或是数据库系统。问题在于记录到 syslog 的日志消息只是由 IPS 生成的警报摘要信息，它丢失了重要的信息，你可能需要去数据库获取完整的数据集。例如，数据包捕获（PCAP）。PCAP 包含触发 IPS 生成警报或者日志消息的网络连接部分。这可能是用于信息分析、报告的一个重要的项目。

1.2.3　什么是日志消息

正如我们之前讨论过的，日志消息是由某些设备或者系统生成，用来表明发生某些事情的消息。但是日志消息是什么样子的呢？让我们来简短地看一下这个问题的答案。本书将会非常详细地介绍这个内容。

理解这个问题的答案，是本书剩下部分的基础。

首先，典型的日志消息的基本内容如下：

- 时间戳
- 源
- 数据

不论消息是通过 syslog 发送、写入 Windows 事件日志或者是存入数据库，上述基本项目总会是消息的一部分。时间戳指示了生成日志消息的时间，源是生成日志消息的系统，它通常是以 IP 地址或者主机名的形式表示。最后，数据是日志消息的核心内容。遗憾的是，日志消息中的数据并没有一个标准的格式。你会在一条日志消息中发现一些比较常见的数据项，包括源和目标 IP 地址、源和目标端口、用户名、程序名、资源对象（如文件、目录等）、传入或者传出的字节，等等。

就像我们之前提到的，具体的日志消息表现形式取决于它的来源如何实现日志数据系统，syslog 是在设备和计算机系统中使用的最常见的格式。因此，让我们来看一条 syslog 消息的例子。图 1.1 给出了一个例子让我们思考。

```
        1                2                          3
Jul 13 19:17:23 10.240.46.16 15: *Mar  1 00:16:17: %LINEPROTO-5-UPDOWN: Line
protocol on Interface FastEthernet0/0, changed state to up
```

图 1.1　syslog 日志数据的例子

图 1.1 中的示例是运行 syslog 服务器的日志主机接收到的由 Cisco 路由器生成的 syslog 消息。注意，日志消息上面的数字 1、2、3 并不是日志消息的一部分，而是本次讨论中的参考点。参考点 1 是日志主机接收到日志消息的时间戳。时间戳包括收到时的月、日、时、分、秒。参考点 2 是 Cisco 路由器的 IP 地址，注意，IP 地址后有一个冒号。下面我们继续看参考点 3。在 IP 地址和冒号后面的都是日志消息数据，格式很随意，包含了丰富的信息。例如，日志消息的优先级是它所传达的主要信息之一。这是供应商对日志消息优先级（有时候称为严重性）的解读。在图 1.1 中，优先级嵌入在这段字符串中："%LINEPROTO-5-UPDOWN"，数字 5 表示它的严重性。但是 5 代表什么意思呢？在这种情况下，您就需要查询供应商的文档，了解日志消息的格式和消息数据中可能传递的任何相关含义。还应当指出的是，在 syslog 协议中有着优先级的方案。这个概念会在第 2 章进行论述。

现在要注意，路由器的消息数据包含一个时间戳，跟参考点 1 中的不一样。这个时间戳是由路由器生成的，它表明了事件发生的大概日期和时间。syslog 服务器的系统时间和路由器的时间都有可能是错误的。无论如何，时间的同步是很重要的，你应该知道系统时钟是否停止、偏差等等，时钟偏差至少可能导致日志数据的一致性问题，在分析任何问题的时候都不妨做这样的质疑。

你也许会感到疑惑，我们怎么面对日志消息有着自由格式这个现实？下一个小节我们将要讨论日志系统的概念，并简要探讨一些其他的主题。

1.2.4 日志生态系统

之前我们已经在概略地讨论了日志数据和日志消息，现在我们来了解日志是怎么在整个日志记录生态系统中使用的。日志记录生态系统，有时也称为日志记录基础设施，是组合在一起实现了日志数据的生成、过滤、规范化、分析和长期存储各项功能的组件和零件。这个系统的最终目标是能够利用日志来解决问题。而需要解决的问题取决于你的环境。例如，如果你是一个处理信用卡交易的零售组织，就必须遵守各种各样的监管和依从性要求。本节中余下的内容将会在后续的章节中展开讲解。

让我们从一些在开始规划日志系统架构的时候需要考虑的事情，开始我们的讨论。

1. 先说重要的事情：问问题，制定计划

大多数的组织都有一个或很多理由来解释为什么一定要记录日志，很少有人只为了好玩而记录。例如下面的问题：

- 我们必须要遵守什么样的监管和依从性要求？某些法规可能要求你证明自己达到依从性的要求。这种证明可能是以报告形式说明你记录了财务系统、信用卡交易系统等关键资产的访问日志。其他证据可能是由审计人员审计你的环境中的各个组成部分，例如设备配置库、缺陷跟踪系统、源代码控制系统等等。第 12 章和 16 章将会提供一些关于监管需求的细节以及在日志记录上的应用。

- 我们需要满足什么样的长期存储要求？换言之，我们需要将日志数据保存多久？这会极大地影响到存储日志数据的磁盘存储容量。
- 我们是否需要从局域网（LAN）或者广域网（WAN）的设备上收集日志数据？这会影响到从日志客户端到服务器的网络连接和吞吐量。
- 我们收集日志的最终目的是什么？如果你只是对收集 CPU 性能信息感兴趣，那么你的处理方式和监控防火墙完全不一样。尽管监控防火墙的 CPU 性能消耗可能也是你所关注的目标。

2. 日志消息的生成

正如我们之前讨论的，日志是由许多源生成的，例如：

- Unix 和 Windows 操作系统
- 路由器
- 交换机
- 防火墙
- 无线接入点
- 虚拟专用网络（VPN）服务器
- 防病毒（AV）系统
- 打印机

这个列表还可以延续下去。这里要说明的是，几乎所有的设备、计算机系统和你网络中的应用程序都是能够生成日志的。你只需要知道去哪里找到它们。但是除了寻找之外，你还需要配置你的源系统，以便真正地记录日志。接下来是在大部分设备和系统上开启日志记录的三步要点：

1）启用设备的日志记录。通常你会发现可以开启或者关闭日志记录。

2）配置它来发送日志消息。一些系统只能开启或者关闭日志记录。在这种情况下，这一步可以省略。其他系统允许你精确调整所记录的日志，例如，尽量不影响源系统上的资源。

3）配置它把日志发送给日志主机以便于更进一步地收集和分析，这一步是不言自明的。

3. 日志消息的过滤和规范化

一旦设备或者计算机系统被配置为生成日志消息，下一步就是过滤和规范化这些消息。过滤就是根据内容对日志消息做包含或者排除（排除在某些情况下等同于“丢在地上”或者抛弃）的处理。有些源本身就提供支持，或者在某些情况下，你可能要使用代理，实际上就是拦截日志消息并且根据用户定义的规则来过滤它们。决定怎么过滤在极大程度上取决于所在组织的需求。例如，在常规维护窗口丢弃 Cisco 路由器的重启信息是完全合理的。

规范化就是将不同格式的日志消息格式化，并且将它们转换成通用格式的操作。“事

件”这个术语通常用于表示规范化的日志消息。事件通常是中间状态的日志消息，可以把它看作是日志记录系统中流通的货币。不过，通常情况下，事件最终存在的地方是关系型数据库，在那里将会进行分析和报告。如果你的日志数据是通用格式的话，操纵数据并从中获得信息将会方便得多。需要说明的是规范化的进行通常是不管使用的源或协议的（例如 syslog、SNMP、数据库，等等）。

在看一些简单的例子前，我们来讨论一下优先级。正如我们之前提到的，有些日志消息本身就有着优先级，而有些日志消息可能没有。优先级的概念也是需要规范化的。一个基本的方案就是获取消息的明确和隐含优先级，并将其映射到一些通用方案中。例如低、中、高级别。如下描述只是基本的例子，仅仅给你提供一个起点：

- 低级：低级事件是发生时不需要处理的某些信息。
- 中级：中级事件往往是一些需要按时查看，但并不需要即时处理的事情。例如，IPS 往往有能力在引擎检测到恶意请求时拦截网络流量。拦截网络流量指的是一种操作。如果你的 IPS 对这次入侵记录了日志消息，那么你知道流量被拦截了，可以在不忙的时候来调查这个事情。
- 高级：高优事件是需要马上进行干预的，例如路由器在批准的维护窗口之外的重启、IPS 引擎对于信息可能泄漏的警报、网络设备断网时间超长等。

但是，规范化是如何完成的？第 9 章将会详细介绍规范化，不过我们先来看一个简单的例子。下面是一个 Sourcefire IPS syslog 消息：

```
Jul 16 10:54:39 SourceFire SFIMS: [1:469:1] ICMP PING NMAP
   [Classification: Attempted Information Leak] [Priority: 2] {ICMP}
   210.22.215.77 -> 67.126.151.137
```

这条消息里面有大量的信息。为了将其规范化，我们使用一种叫做解析（Parsing）的技术。解析需要从头到尾的扫描整个日志消息，将其中我们感兴趣的部分抽出来放在事件的规范化字段中。下面就是一组用于规范化的通用字段：

类型：企图泄露信息

时间戳：2010 年 7 月 16 日，10:54:39

优先级：高

协议：ICMP

源 IP 地址：210.22.215.77

目标 IP 地址：67.126.151.137

源端口：无

目标端口：无

原始日志：Jul 16 10:54:39 SourceFire SFIMS: [1:469:1] ICMP PING NMAP [Classification: Attempted Information Leak] [Priority: 2] {ICMP} 210.22.215.77 -> 67.126.151.137

下面是一些说明：

- 源和目标 IP 地址非常容易识别，一个箭头（->）用于表明可能泄漏的方向。
- 我们将优先级定为“高”，因为网络传输并没有被拦截，所以我们需要马上查看这个企图。
- 源和目标端口并不是日志消息的一部分，所以它们都被置为空。
- 原始日志包含了实际的原始日志消息。这有利于保存原始消息。

让我们来看看 Snort 生成的另一条消息：

```
Mar 12 18:02:22 snort: [ID 702911 local4.alert] [119:2:1]
   (http_inspect) DOUBLE DECODING ATTACK {TCP} 10.1.1.21:60312 ->
   10.1.1.69:80
```

我们很幸运地发现，这个格式跟 Sourcefire 的那条很相似，理由很充分——Snort 是 Sourcefire 的开源版本。现在注意，源和目标 IP 地址后面都跟着一个冒号以及一个数字。这些数字分别表示着源和目标端口。而且，这条日志消息跟 Sourcefire 的那条不一样，并没有优先级的字段。这就使映射优先级变得更加困难。我们需要对这条消息所包含的信息了如指掌，这就意味着我们需要参考这一警报的相关文档。

现在我们来看看收集日志消息意味着什么。

4. 日志消息的收集

图 1.2 展示了在日志收集背后的基本概念。

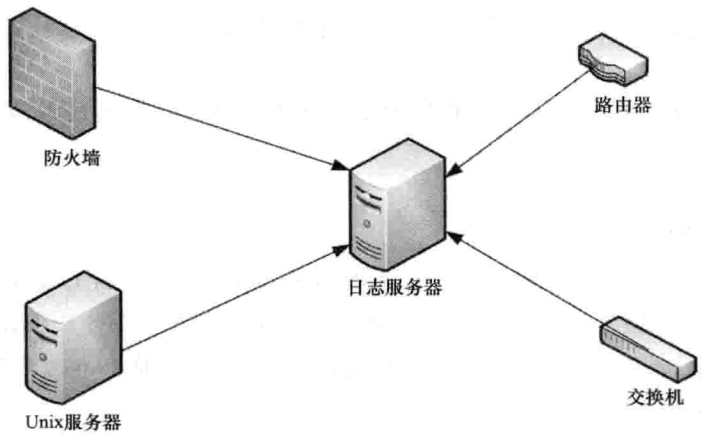


图 1.2　设备日志记录服务器收集数据的逻辑布局

许多不同种类的设备和系统将他们的日志消息传递给日志服务器。日志服务器可能存在于一个特定的地方，作为组织中唯一的服务器提供。在组织很小或者地理位置不太分散的情况下，这样做是没有问题的。但是在可能跨越几个州甚至国家的多场地情况下，你可能需要一组分布式的服务器，图 1.3 展示了这种特别的配置。

这里我们将站点特定日志收集器的数据转发给集中日志服务器。因为日志收集器和日志

服务器不在一起，为了防止被窃听，它们之间的通信一般是加密传输的。当日志服务器是集中收集点的时候，它也可以收取日志消息，就像图 1.3 中的 Windows 服务器场一样。

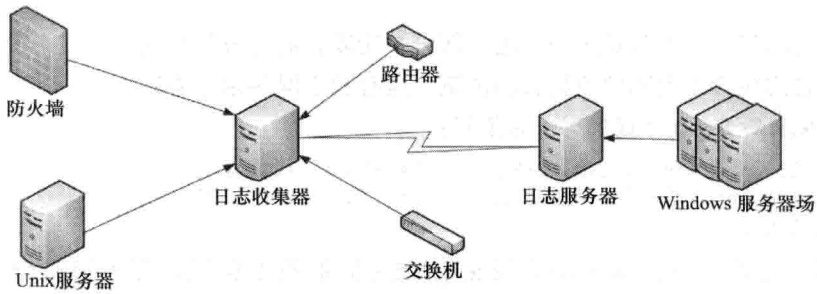


图 1.3　分布式日志记录安装的逻辑布局

这种配置有如下一些优点：

- 冗余：日志数据存储在多个地方。
- 存储转发：如果日志收集器和日志服务器之间的连接断开了，日志收集器可以将日志存储在本地直到重新连接成功再进行传输。这会降低数据丢失的风险。
- 认证：不仅日志收集器需要验证发送者为可信源，发送者在发送日志的时候信任日志系统也很重要。
- 私密：记录日志的设备和系统往往没有加密日志消息的能力。由于日志收集器是在本地局域网内，未加密的数据发送给本地收集器，它有能力将日志以安全的方式发送出去。

我们已经介绍了关于日志数据收集的基础知识，现在让我们简短地讨论云中的日志记录。

5. 在云端记录日志

最近有很多关于云计算的新闻。Google 让你可以在它们的云端编辑和共享文档，Microsoft 也有类似的服务。但是在云端记录日志怎么样呢？有一些组织例如安全托管服务提供商（MSSP）和云日志公司允许你将日志指向它们，由它们完成剩下的工作。图 1.4 展示了在云端记录日志的概念。

在云端记录日志的时候到底发生了什么呢？包括一些过滤、规范化、分析和长期存储之类的事情。云日志服务供应商根据不同的定价来提供服务。它们管理复杂的传统日志收集系统。在第 21 章将更详细地介绍关于云日志的内容。

6. 日志分析

日志消息分析（或者简称日志分析）处理分析日志数据，并从中得到它的含义。回忆一下图 1.3 的集中收集服务器。集中收集点对于分析很关键，把你的数据放在同一个地方，就

可以把日志消息放在一起或者关联起来，以便分析其意义。这在高度分布式的环境中尤为重要，在这种环境中你可能拥有很多远程日志收集器，必须将一个日志收集器收集到的日志消息与其他收集器收集到的日志消息关联起来。第 9 章将会围绕关联和分析作更详细的说明。

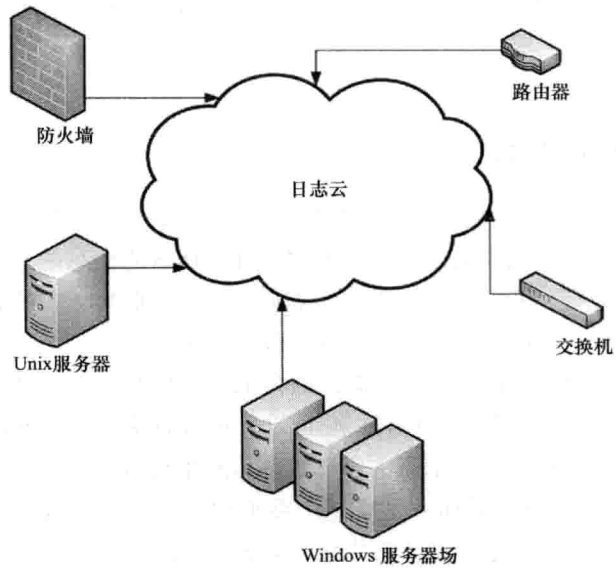


图 1.4　云日志的逻辑布局

7. 日志消息的长期存储

长期存储是讨论的最后一个主题。图 1.5 展示了将数据库添加到日志服务器。

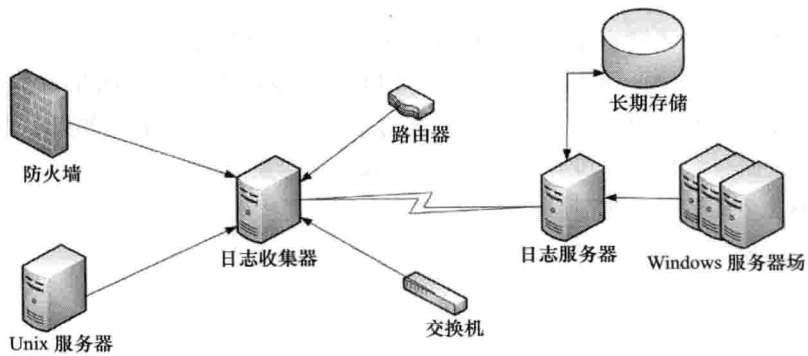


图 1.5　为长期存储添加数据库

数据的长期存储通常是由关系型数据库完成的。在远程日志收集器中使用小型数据库也很常见：

- 报告：为管理员或者依从性审计生成总结性的日志数据报告。
- 取证分析：你可能需要看看一个月前一个特定的用户在网络中做了什么。
- 备份：这一点不言自明，但是如果保存日志数据是由于监管依从性等要求，你的数据备份永远都不嫌多。

现在你已经很快地过了一遍日志记录生态系统中的各个要素，第 7、15 和 20 章将更详细地介绍本节提供的概念。

1.3 看看接下来的事情

下面我们来简单看看你将会在本书的剩余部分遇到什么样的事情。由于必须要有日志信息，你才能从中获取到有用的信息，因此我们将关注生成日志信息的一些事情，以及如何让它们提供给你所需的东西。我们将会向你展示如何构建一个收集、归档和分析日志数据的基础设施。在这个过程中，我们会提供真实的例子和案例研究。

我们将会详细讨论 Unix 和 syslog 数据，因为很多应用程序默认使用 syslog，很多其他的应用程序也对其兼容（例如大部分 Cisco 和 Juniper 的设备都使用 syslog）。并且，syslog 是唯一具备向集中收集系统转发日志消息的内建机制的跨平台日志记录工具。但是，我们也会讨论在你的系统中可能使用的其他形式的日志。我们也肯定会谈到 Windows 的日志记录，主要是关于将你的 windows 日志数据放入集中化基础设施的内容。

如果你是程序员，我们专门为你准备了一整章。坦率地说，分析日志消息最大的一个问题就是大部分日志的质量低下，没有包含任何可用于有用操作的信息。如果从一开始就提高日志质量，它的使用就会简单得多。我们来看看下面这条日志消息的例子：

```
Jun 21 14:38:25 10.2.2.1 rlogin: connection refused
```

这条消息有什么问题吗？它很能说明问题，不是么？它告诉我们有一个向远程登录服务的连接被拒绝了。除了没有年份、没有时区这些令人烦恼的事情之外，如果它能告诉我们从哪里发出的连接被拒绝，会不会更好一点？如果告诉我们连接被拒绝的原因会不会更好一点？如果你只是在诊断和测试，那么这条日志消息并不是那么糟。但是如果你正在为某些人攻击了你的系统而查看日志，知道攻击从哪里发起显然会更有用。

现在来考虑下面这条日志消息：

```
Jun 21 14:38:25 somehost rlogin: connection refused from 1.2.3.4
   because the address matches deny rule in hosts.allow: rlogind: all:
   deny
```

是不是好了很多？

下面是一条实际上没什么用处的消息（这可不是我们编造出来的）：

```
May 21 00:00:07 baz.example.com kernel: "00 00 00 00 00 00 00 00 00 00
   00 00"
```

也许它对内核开发者的确有某些意义，但在其他方面没有什么帮助，它就跟下面的日志一样：

```
May 21 00:00:07 baz.example.com kernel: something happened
```

在我们为程序员准备的章节中，并不是仅仅抱怨这些糟糕的日志消息，而是对如何提高应用程序生成的日志质量提供很多有用的建议。

1.4　被低估的日志

在很多企业环境中，日志没有得到重视。日志往往在日常工作中被完全忽视，仅仅在磁盘空间不足的时候才会引起人们的注意。而在这个时候它们往往未经查看就被删除了。某些情况下，日志中的一些消息可能指出磁盘满的原因。我们肯定都有过查看已被入侵的机器的经历，在询问日志保存的位置之后，我们会听到："噢，它们只会占据空间，所以我们把它们删掉了。"在大多数这种情况下，我们没有什么可做的。

为什么日志不受重视呢？这是有很多原因的。供应商并不希望你使用它。入侵检测系统的供应商会告诉你需要最新的科技（他们是最好的），否则你就会完蛋。IT 管理工具供应商会告诉你需要他们的高价产品，只要在每台主机上安装少数代理，就可以获得和日志相同的信息报告。而如果你可以从日志中获得信息，当然就不需要他们的产品。

即使对系统管理员来说，日志也不"性感"。Gigawombat-3000 网络入侵检测系统（NIDS）具有 Extra Sensory Perception（超感官洞察力），在攻击发生之前就能预知，看上去比日志要帅气得多。当你最喜欢的邮件列表源源不断地向你发来关于入侵检测 / 预防 / 预测 / 等方面的最新信息时，这一点更加明显。顺便说一句，你买的 Gigawombat 也需要有人来分析它的日志。

日志分析并不简单，可能相当棘手。和"插入然后祈祷"的方法相反，有效的日志分析有一定的效果。日志以各种形状和大小出现，有时候很难从中提取信息。syslog 数据可能相当糟糕，因为大多数数据都是自由格式的文本（正如我们之前讨论的）。正如我们即将在本书中向你展现的，从 syslog 中获取有用的数据需要花费一些精力，而且需要处理的数据量可能很大。例如，有些网站每周会收集几个 GB 的日志数据，有的可能一天内就能达到这样的量级。这样的数量似乎令人不知所措，管理员最终往往根据特定的时间内看到的东西，人工拼凑出脚本来寻找一些随机的东西。

有效的日志分析还需要了解环境。你必须知道，什么对你的网络有利，什么对你的网络有害，什么是可疑的，什么是正常的。对你有害或奇怪的东西很有可能对别人是很正常的。例如，一个用户从外国登录可能对你来说很可疑，特别是你所有的用户都是本地的情况下。然而，另一个网站的用户可能遍布各地，所以他们很难确定什么才算是不寻常的。

这就是没有真正即插即用的日志分析工具的部分原因，因为你的环境和策略决定了希望从日志中获取的信息。

最后，关于如何做日志分析并没有很多有用的信息。嘿嘿，也许应该有人写一本这方面的书啊。

1.5 日志会很有用

除了到目前为止本章展现的许多例子之外，日志还能告诉你很多关于网络中所发生事件的信息，包括性能信息、故障检测和入侵检测。日志会成为在事故发生后查明“发生了什么”的一个很好的“取证”信息来源。日志可以为审计进行审计跟踪（还有什么？）。

以下小节列出了日志的一些用例。

1.5.1 资源管理

日志中包含了丰富的隐藏财富。例如，监控一台主机是否在线的典型方法之一是使用Internet 控制报文协议（ICMP）来“ping”主机。但是，这给出的信息并非很准确。成功ping 一个主机只能说明它的网络接口配置没有问题。但有时一台主机可能已经崩溃，而此时只要它配置好并且有电，接口就能响应。看看这样的消息：

```
May 24 02:00:36 somehost -- MARK --
May 24 02:20:36 somehost -- MARK --
May 24 02:40:36 somehost -- MARK --
May 24 03:00:36 somehost -- MARK --
```

这些消息告诉你系统正在运行，syslog 可以写入消息。（MARK 消息是一种 UNIX syslog 守护进程定期生成的特殊状态消息。）

日志不仅仅可以告诉你主机是否运行，还能够告诉你主机上运行的应用程序在做什么。往往在系统真正宕机前日志中就展现出了硬件和软件的错误。当从错误恢复的时候，日志往往提供了导致故障的线索。

下面是一个在日志中找到故障的经典例子：

```
May 21 08:33:00 foo.example.com kernel: pid 1427 (dd), uid 2 inumber
   8329 on /var: filesystem full
```

这些信息相当有用！它说明主机上的 var 文件系统已经用满。消息中显示的另一个信息是导致这个错误发生的进程的名字和进程 ID。它甚至显示被写入文件的 i 节点。在这种情况下，显示的进程可能并不是填满磁盘的那一个，它可能只是在分区已被填满的情况下试图写入数据。但这是一个起点。

1.5.2 入侵检测

主机日志（不同于 NIDS 日志）对入侵检测当然是非常有用的。例如：

```
Sep 17 07:00:02 host.example.com: sshd[721038]: Failed password for
   illegal user erin from 192.168.2.4 port 44670 ssh2
```

这条日志消息展示了一次使用用户名“erin”的失败登录尝试。“illegal user”（非法用户）一词的使用是因为系统上并没有这个账户。这条消息是一个攻击者使用 ssh 扫描器，试图用一组常见的用户名和密码通过 ssh 登录主机的结果。本例子是从很短时间内发动的数千次攻击中选出来的。

主机日志可能是比 NIDS 更好的入侵指示器。NIDS 能够告诉你针对主机发生了一次攻击，但是不能告诉你攻击是否成功了。在上面的例子中，NIDS 只能检测到在很短的时间内可能发生了大量 ssh 会话，但是不能检测到认证已经失败，因为 NIDS 只能知道线路上发生了什么。如果配置了主机日志，可以提供系统上所发生情况的细节。当然这也不是万无一失的。你可能会记录下错误事件或者磁盘满的情况。攻击者也有可能擦除日志文件。这就是将日志发送到远程收集点的重要性所在，哪怕攻击者掩盖了他的痕迹，我们在日志收集点上也有原始日志消息，能向我们揭示可能发生的情况。来看一个更微妙的例子，下面是一条来自 Snort（一个开源 NIDS）的消息：

```
Jan 2 16:19:23 host.example.com snort[1260]: RPC Info Query: 10.2.3.4
   -> host.example.com:111
Jan 2 16:19:31 host.example.com snort[1260]: spp_portscan: portscan
   status from 10.2.3.4: 2 connections across 1 hosts: TCP(2), UDP(0)
```

这些消息表明攻击者对网络做了一次端口扫描，试图寻找运行 rcp.statd 的主机，这一服务近年来发现了大量易受攻击的漏洞。消息还表明扫描器成功连上了两台主机。但是攻击成功取得了进入机器的权限了吗？连接是否被 TCP Wrappers 断开了呢？ Snort 不知道。不过，运行 TCP Wrappers 的系统更可能记录了连接是否断开的日志。如果该系统被设置成记录日志，你就能够把 Snort 日志消息和 TCP Wrappers 的消息关联起来、互为补充，更好地展示实际发生的情况。永远不要低估来自多个源的日志消息相互结合、描绘更翔实情况的能力。

下面来看看在一台主机上找到的如下日志：

```
Jan 02 16:19:45 host.example.com rpc.statd[351]: gethostbyname
   error for ^X÷ÿ¿^X÷ÿ¿^Y÷ÿ¿^Y÷ÿ¿^Z÷ÿ¿^Z÷ÿ¿^[÷ÿ¿^[÷ÿ¿bffff750
   8049710909090906874656762747 36f6d616e797265206520726f7220726f66
   bffff718 bffff719 bffff71a bffff71b_____________________
   ____________________________________________________
   ____________________________________________________
   ____________________________________________________!
__!
_____________________________________________________________________
   ___________
```

唔，看起来这很像是一次缓冲区溢出攻击的尝试，或者最起码发生了某种错误，因为在第二行开始的乱码向我们说明了这一点。乱码表明存在二进制数据，即不可打印字符，这在日志消息中通常是一个不好的标志。

但是攻击是否成功了呢？下一条消息给了我们线索：

```
Jan 02 16:20:25 host.example.com adduser[12152]: new user: name=cgi,
   uid=0, gid=0, home=/home/cgi, shell=/bin/bash
```

这条消息说明系统中添加了一个账户。该账户的用户名是 cgi，而用户 ID 是 0（根 uid）。这通常是一个不好的兆头。攻击者喜欢创建账户，以便用于登录机器。他们也很喜欢选择在系统中容易被忽视的正常用户名，即那些似乎本来就应该存在的名称。但是入侵者想保留他的访问特权，所以账户有跟 root 一样的 uid（0）。

缓冲区溢出

缓冲区溢出是一种利用应用程序缺陷的攻击类型。攻击者利用应用程序漏洞，在程序中输入超出处理能力的数据。额外的数据注入可能产生崩溃等意外的副作用，在极端情况下，作恶者可能获得系统处理器的控制权，用于执行任意代码（Peikari & Chuvakin,2004）。

下一条消息是：

```
Jan 02 16:22:02 host.example.com PAM_pwdb[12154]: password for (cgi/0)
   changed by ((null)/0)
```

这行日志中最有趣的部分是，它指出某个用户的密码被名为 null 的用户修改了。这也是一个很糟糕的迹象，使用正常方式登录的用户会有一个实际的用户名（在本例中应该是“root”）。

所以，在上面的例子中，NIDS 并不能够告诉我们攻击是否成功，只能告诉我们可能有人试图攻击。真正的攻击信息被记录在主机上。NIDS 擅长告诉你查看日志的好时机，但是仅仅靠它是无法提供完整的情况的。当然主机日志并不是总能说明发生了什么，但是很明显，将 NIDS 和日志结合起来比 NIDS 本身的作用更大。想起引言中的例子了吗？在大海捞针的过程中，永远不要低估将日志数据结合到一起的威力。

日志告诉你，系统中到底发生了什么。NIDS 往往不能告诉你用户尝试利用主机的本地漏洞，也不能说明用户违反了策略或者进行其他不允许的活动。主机入侵检测系统（HIDS）填补了这方面的空白。HIDS 监控计算机系统的运行状态，通过校验和（Checksum）以及其他的一些技术来监控用户、目录、二进制文件、日志文件本身以及其他一些对象，检测它们何时遭到恶意用户、遭到入侵或者修改的应用程序的篡改。你可能已经猜到，HIDS 系统在检测到系统资源修改的时候生成日志。

随着安全套接字层（SSL）、Internet 协议安全（IPSec）等加密传输协议和其他的 VPN 技术越来越多地得到采用，NIDS 越来越难以发现网络流量中“有趣”的部分。

正如前文提到的，用户行为也包含在系统日志中，当一个用户登录、注销或者从某处进入等情况发生时，有些日志（如进程账户日志）会告诉你一些关于用户行为的信息。系统“审计”工具（如 BSM）能够提供粒度更细的用户和系统活动详情。

1.5.3　故障排除

日志对故障排除也很有价值。事实上，syslog 就是为了这个目的而设计的。如果不检查日志文件，诊断没有连接到控制台或者 tty 的守护进程是相当困难的。而且系统内核通常通过 syslog 来提供与其所做的事情相关的信息。

1.5.4　取证

取证是在事件发生后重建“发生了什么”的情景的过程。这种描绘往往基于不完整的信息，而信息可信度是至关重要的。日志是取证过程中不可或缺的组成部分。

日志一经记录，就不会由于系统的正常使用而被修改，这意味着它们是一种“永久性”的记录。因此，它们可以为系统中其他可能更容易被更改或破坏的数据提供准确的补充。

由于每条日志中通常都有着时间戳，它们提供了每个事件的时间顺序，不仅仅告诉我们发生了什么事情，还告诉我们这些事情发生的时间和顺序。

而且，日志发送到另一台主机（通常是一个集中日志收集器），这也提供了独立于原始源的一个证据来源。如果原始来源上信息的准确性遭到质疑（例如入侵者可能篡改或者删除了日志），独立的信息源可能被认为是更可靠的附加来源。

同样，来自不同来源甚至不同站点的日志可以佐证其他证据，提高每个源的准确性。

日志有助于加强收集的其他证据。重现事件往往不是基于一部分信息或者单个信息源，而是来自各种来源的数据：文件和各自系统上的时间戳、用户的命令历史记录、网络数据和日志。日志有时可能会否认其他证据，这本身可能表明其他来源已被破坏（例如遭到入侵者的篡改）。

日志如何抓住入侵者

几年前，某公司的 Web 服务器崩溃了。从 Web 服务器的日志上看，似乎发生了一次利用了最近公布的拒绝服务（Dos）漏洞的攻击。连接的来源是同一个城市的另一个公司，这时 IT 部门的人想起，最近离职的一位员工已经开始为另一家公司工作，这位员工在离职时没有得到最好的补偿。现在他们的日志仅仅说明攻击是来自另一家公司，但是被告攻击者可以声称该日志是经过伪造、用来“陷害”他们的。然而，除了 Web 服务器的日志外，两家公司的防火墙日志显示出在相同的时间有相同的连接。来自攻击者所在新公司的日志表明，攻击来自于他的桌面电脑。最致命的证据是，受害公司的一位员工致电这位前雇员的新单位，而他接听了电话，说明在攻击的时候，他正好在自己的电脑前。攻击者最终承认了所犯的罪行。

如果受害公司只有自己的 Web 服务器日志，他们可能就没有足够的证据来支持诉讼。拥有多种来源的数据支持，尤其是在不同的群体控制下的来源，就能够更好地对抗不正确或者伪造证据的指控。

日志中显示的证据有时是间接或者不完整的。例如，一个日志条目可能展示出一个特定的活动，但没有说明是谁进行的。或者，举个例子，进程账户日志显示一个用户运行了什么命令，但并没有记录这些命令的参数。所以日志并不总是唯一的可靠信息来源。但是，当主机遭到入侵，如果主机已经将日志转发给集中日志服务器，日志就可能是唯一可靠的信息来源。这些日志在主机遭到入侵前是可以信任的，而在入侵之后就值得怀疑了。但是你收集的日志有助于揭示到底发生了什么，或者为你指出正确的方向。

1.5.5 无聊的审计，有趣的发现

审计是验证系统或者过程是否如预期般运行的过程。日志是审计过程的一部分，形成审计跟踪的一部分。

审计往往是为了政策或者监管依从性而进行的。例如，公司往往需要做财务审计，以确保他们的财务报表和账簿相符，且所有数字都合情合理。Sarbanes-Oxley（萨班斯 - 奥克斯利法案）和 HIPAA（健康保险便利性与责任法案）等美国法规都要求某种交易日志，以及可以用来验证用户对金融和患者数据的访问的审计跟踪。另一个例子是 Payment Card Industry Data Security Standard（PCI DSS，支付卡行业数据安全标准），它的强制要求包括记录信用卡交易日志和持卡人的数据访问日志。我们将在 16 章讨论 PCI。

日志也可以被用于遵从技术策略（如安全策略）的依从性。例如，如果你有一些在你的网络中允许使用哪些服务的策略，可以采用对各种日志的审计，来验证只有这些服务在运行。例如，Windows 有一个应用程序事件日志，包含了操作系统启动或者停止服务的日志消息。

审计跟踪也可用于证明可审核性，日志本身可以用于防抵赖。例如，如果有人声称他们从来没有接收一个特定的邮件，邮件日志可以用于核实并显示邮件到底有没有发送，就像邮件投递员签收的单据一样。再举一例，如果发生某一事件时，一个用户声称他无法登录，日志能证明他们是否说了实话。

有条理地进行日志分析，还能把一个无聊的审计变为“有趣的发现”。这是什么意思呢？例如，我们将会为你展示一些工具，你可以用它们来从日志中获取主机上运行的所有服务的列表。你也许会吃惊地看到正在运行的一些服务（你知不知道网络上有个 IRC 服务器？）。有时，这些发现证明有人违反了策略，有时，它们说明有一位入侵者愚蠢地启动了一个留下日志记录的服务。

审计用户活动可以展示一些你不知道的事情。在 Unix 环境中，有一个叫 sudo 的实用程序。它能允许一个普通的用户去执行管理员的命令而不需要知道管理员的密码。查看 sudo 的日志，可以告诉你谁运行了管理员命令。如果有人无意中删除了整个文件系统，而你想知道是谁做的，这个信息就很有价值。

如果你允许用户从远程登录，也许会对他们从哪里登录感兴趣。如果你发现一个用户从两个不同并且有很大距离的地点同时登录，也许就有必要去调查了。（我们听说手机运营商就

是这样来侦测克隆手机的）。

下面我们把重点从日志的重要性转移到人、过程和技术在整体日志分析中的效果上来。

1.6　人、过程和技术

有效的日志分析策略不单单是一系列的工具，而是人、过程和技术的完美组合。

技术是你使用的各种工具的组合。但是只有最新的工具是不够的，你必须知道用它们来做什么。

过程决定了你如何使用这些工具。如何管理日志数据，确保它含有你所需要的信息，并从中获取你所需要的信息？过程对于这些问题来说是必需的。

在法庭上用日志作为证据时，一个记录在案的日志处理过程是必不可少的。能够展示你平时怎样收集和保存日志数据，会极大地影响到你的日志数据是否可以成为证据。报告可以对此提供帮助，我们会在第 12 章涉及这个方面。

过程的推动和执行需要人。确定特定的日志消息在特定情况下的含义也需要人的智慧。

上面的入侵案例就是一个需要人类参与的例子。计算机并不知道那些日志记录就意味着入侵。需要一个人去查看日志，然后提出“等一下，好像有点问题，为什么一个账户的 uid 是 0？”这种情况的另一个例子是突然无法连接上一个主机的时候。因为你在早上的会议中宣布网络维护将在下午三点进行，可以利用这个信息，判断连接中断是否与此有关。

1.7　安全信息和事件管理（SIEM）

日志管理的书籍如果不讨论 SIEM，那就是不完整的。SIEM 已经形成了一个行业，SIEM 工具提供了一种实时分析安全事件的方法。它也提供了报告、可视化和长期存储等机制。我们在这本书里面不会花太多时间在 SIEM 上，但是第 15 章将会讨论一些开源的 SIEM 工具。

下面的两个案例研究会解释日志数据怎样帮助解决现实世界的问题。

通过防火墙日志跟踪攻击者

在最近的调查中，本书的一位作者需要查看在小型互联网服务提供商（ISP）运行的一台遭到入侵的服务器。该系统明显被攻击者摧毁了，这很有可能是因为攻击者无法进一步渗透到网络中，或者是由于攻击者的险恶用心。

一天早上，公司支持团队收到一位尝试下载驱动程序更新的客户的警报，这位客户报告说 FTP 服务器对他的连接尝试“没有响应”。在通过安全 Shell 程序登录 FTP 服务器失败后，支持团队成员进入机房，发现机器已崩溃无法重新启动。原因很简单——在磁盘上

找不到操作系统。

机器自身明显受损，所有主机端证据（包括日志）都不翼而飞（是的，日志或许可以用“取证”工具恢复，但那超出了本书讨论的范围）。但是服务器所在的网段处于防火墙之后，而且，防火墙日志设置为最高级别。

支持团队在防火墙日志中搜索了过去24小时内包含受攻击服务器IP地址的任何消息，提供了前一天服务器所有出入连接的记录。通过这些日志可以确定：

- 服务器崩溃前数小时，有人从外部探查FTP服务器的IP地址。这些连接尝试了服务器的多个端口，其中有不成功的（如TCP端口25），也有成功的（TCP端口21，即FTP）。
- 攻击者曾多次连接到FTP端口，连接时间从数秒到十分钟不等。
- 攻击者曾上传一个文件到FTP服务器。
- 攻击者似乎设法建立了与服务器的一个交互式命令会话。防火墙日志并没有给出关于这一TCP会话内容的细节，所以无法确定攻击者做了什么。但是他极有可能利用FTP服务器在服务器上安装了自己的软件。
- 之后的日志说明，FTP服务器尝试连接外部主机，这些尝试遭到了防火墙限制出站通信规则的拦截。这些失败的尝试很有可能是入侵者通过被入侵的服务器尝试访问其他机器时发生的。

所以，即使是简单的防火墙日志，也能在没有其他有用信息的情况下，揭示事故的重要信息。这个例子也可以从中央日志主机得到好处。这些日志会被储存在主机之外，可以被恢复。

通过防病毒日志跟踪病毒

防病毒日志常常被看作平凡而无用的。然而，防病毒日志可以证明，它作为确定系统是否感染或受到入侵的证据，是很有价值的。不同的防病毒供应商用不同的方法记录他们的消息。这个例子中使用的防病毒数据是由防病毒系统记录在Windows事件日志中的。

在一个案例中，Windows系统有不寻常的表现：

- 系统对本地程序的响应很慢。
- 在同一LAN网段中，网速很慢，就像系统存在问题的时候一样。
- 即使没有用户在使用系统，依然有外部进程在运行。

来自网络安全监控工具的日志也说明，该系统是大量扫描活动的来源。在那个阶段，机器从网络中断开。

系统管理员曾怀疑有某种恶意软件（如病毒，蠕虫等），但系统正运行着防病毒程序，而且这个程序必定是包含了最新病毒库的。

对系统防病毒日志进行审核，揭示了大量有趣的现象。首先，一些消息指出，失败的更新是因为磁盘空间不足。这就解释了为什么系统没有得到全面的保护。

其次，有些消息表明防病毒软件对病毒做出了响应：

```
Virus Found!Virus name: W32.Welchia.B.Worm in File: C:\WINNT\
   system32\drivers\svchost.exe by: Manual scan. Action: Quarantine
   succeeded:
Virus Found!Virus name: W32.Randex.gen in File: C:\WINNT\system32\
   wumgrd.exe by: Manual scan. Action: Quarantine succeeded:
```

以下消息表示发现并“隔离”或阻止了病毒：

```
Virus Found!Virus name: Backdoor.IRC.Bot in File: C:\WINNT\system32\
   mfm\msrll.exe by: Manual scan. Action: Clean failed: Quarantine
   failed:
```

但是这个消息只显示检测到一个名为“Backdoor.IRC.Bot”的“后门”，并没有停止它。

注释　后门

后门（Backdoor）是允许用户在不为人知的情况下获得计算机系统访问权的软件或应用程序。它经常被用于恶意目的（Skoudis & Zeltser,2004）。

上面的消息中，有趣的是虽然两个恶意软件得到处理，但最后一个只是被检测到，并没有被清除。这个恶意软件（一个 IRC 后门程序）使攻击者控制系统，并用它来扫描其他主机的漏洞。

图 1.6 显示了一个导入到 Excel 的防病毒日志子集，说明了对 Backdoor.IRC.Bot 采取的行动。Excel 在这里可以更清晰地展示 AV（防病毒）日志消息。

	A	B	C	E	G	H	J	M
1	Date	Filenam	Virus Name	Action Take	User	Original Location	Current Location	Scan Type
2	7/2/2004 1 07	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Scheduled scan
4	7/1/2004 1 07	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Scheduled scan
5	6/30/2004 19 45	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Realtime scan
6	6/30/2004 19 45	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Realtime scan
7	6/30/2004 19 44	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Realtime scan
8	6/30/2004 1 07	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Scheduled scan
9	6/29/2004 18 29	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Manual scan
16	6/28/2004 21 55	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Manual scan

图 1.6　防病毒日志样例

上述例子说明，这个后门至少从五月份就开始存在于系统里。这也证实了后门没有被防病毒软件所隔离。

在图 1.7 中高亮显示的消息表明，后门大概出现在 5 月份，系统被 Welchia 和 W32.Randex.gen 蠕虫感染，这两个蠕虫都有可能安装该后门。实际上，根据 W32.Randex.gen 的参考资料（http://securityresponse.symantec.com/avcenter/venc/data/w32.randex.gen.html#technicaldetails），这种蠕虫以“打开后门端口和打开到预定的 IRC 服务器的连接，等待攻击者的命令”而著称。

上面的日志条目也表明，该系统是恶意软件的“鼠窝”，只能成功地清除一些恶意软件（但是通常是在成功地感染和修改系统之后）。

所以，在这种情况下，如果没有防病毒日志，我们很难或者不可能确定感染是什么时候

发生，或者防病毒软件在何时更新失败。

	A	B	C	E	G	H	J	
1	Date	Filename	Virus Name	Action Taken	User	Original Location	Current Location	Scan T
14	6/4/2004 20:23	wumgrd.exe	W32.Randex.gen	Left alone in Quarantin	SYSTEM	C:\WINNT\system32\	Quarantine	Defaut
15	6/4/2004 20:23	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\WINNT\system32\drivers\	Quarantine	Defaut
16	5/28/2004 21:55	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Manual
17	5/28/2004 21:51	wumgrd.exe	W32.Randex.gen	Quarantined	Administrator	C:\WINNT\system32\	Quarantine	Manual
18	5/28/2004 19:21	svchost.exe	W32.Welchia.B.Worm	Quarantined	Administrator	C:\WINNT\system32\drivers\	Quarantine	Manual
19	5/17/2004 12:06	25227_up.exe	W32.Sasser.C.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\	Quarantine	Defaut
20	5/17/2004 12:06	25227_up.exe	W32.Sasser.C.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\	Quarantine	Defaut
21	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defaut
22	5/17/2004 12:06	TFTP988	W32.Randex.gen	Left alone in Quarantin	SYSTEM	C:\winnt\system32\	Quarantine	Defaut
23	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defaut
24	5/17/2004 12:06	TFTP1256	W32.Randex.gen	Left alone in Quarantin	SYSTEM	C:\winnt\system32\	Quarantine	Defaut
25	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defaut
26	5/17/2004 12:06	TFTP1128	W32.Welchia.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\	Quarantine	Defaut
27	5/17/2004 12:06	TFTP328	W32.Welchia.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\	Quarantine	Defaut
28	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defaut
29	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defaut
30	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defaut
31	5/17/2004 12:06	WksPatch[1].exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\Documents and Settings\D	Quarantine	Defaut
32	5/17/2004 12:06	WksPatch[1].exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\Documents and Settings\D	Quarantine	Defaut
33	5/17/2004 12:06	WksPatch[1].exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\Documents and Settings\D	Quarantine	Defaut
34	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defaut
35	5/17/2004 12:06	WksPatch[1].exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\Documents and Settings\D	Quarantine	Defaut
36	5/17/2004 12:06	WksPatch[1].exe	W32.Welchia.B.Worm	Left alone in Quarantin	SYSTEM	C:\Documents and Settings\D	Quarantine	Defaut

图 1.7　特定时间段的防病毒日志样例

1.8　小结

这一章提供了一些关于日志记录、日志、日志消息、日志数据的概念。我们讨论了日志记录生态系统的组成部分，包括日志消息的生成、过滤和规范化、收集、分析和长期存储。这些概念为本书剩下的内容打下了基础。附带的例子说明了如何提供有用的日志数据，以及日志帮助解决两个现实问题的方法。

参考文献

Mauro, D. R., & Schmidt, K. J. (2005). *Essential SNMP* (2nd ed.) Beijing: O'Reilly.

Peikari, C., & Chuvakin, A. (2004). Overflow attacks. In *Security warrior* (p. 161). Beijing: O'Reilly & Associates, Inc.

Skoudis, Ed., & Zeltser, L. (2004). Backdoors. In *Malware: Fighting malicious code* (p. 124). Upper Saddle River, NJ: Prentice Hall PTR, 2004.

第 2 章 Chapter 2

日志是什么

本章要点

- 日志的概念
- 良好日志记录的准则

日志是什么？这是一个秘密
让我们来解决它吧！
抵制它是疯狂的
Haiku2:Anton

2.1 概述

在第 1 章我们已经讨论了日志，但是，我们真正讨论的东西是什么？我们正在讨论的并不是树木、数学，或航海日志等等……让我们从定义开始，该领域中的许多讨论是粗略和模糊的，对安全分析人员或网络工程师没有什么用处，因此探寻一个围绕日志记录数据的清晰定义就变得意义非凡了。

定义

在日志记录、日志分析以及日志管理中使用的许多术语（包括我们刚刚使用过的）含义模糊，充满误导或者有多重意义。在某些情况下，术语是从其他学科中“借用”的（我们偶

尔会用到这些词），有时，不同的人使用不同的术语。而在另一些情况下，这些术语曾经很明确，但是被工具供应商的市场部门破坏了。这种模糊性将这一已经难以驾驭的领域变得更加混乱。

那么日志应该传达什么内容？日志应该提醒我们，还是应该被忽略掉？如若日志这般重要，为何它们却常常被忽略（事实上确实如此！）？组织遭到黑客入侵，自己却从不知晓的例子屡见不鲜，而日志中的所有证据都在入侵后存在数月，有时甚至长达数年。如果我们的计算机以及所运行的相关软件、网络设备似乎都正常工作，我们是否还必须查看日志？

理想情况下，日志不应该仅仅作为最重要的“出了什么错”问题的答案。它们还应该用于回答许多其他问题，包括“我们的运行情况如何？”、“未来一段时间是否会出错？”、“对某些事情我们是否做足了保护措施？”，以及大量审计相关的问题，例如“何人做了何事？”，请注意，后者并不仅在我们知道一些错误已经发生的情况下有用。

因此，日志分析工具不应该只在发生错误的情况下派上用场。最终，必须将持续的日志审核培养成一种习惯，就像人们每天刷牙一样。现在，许多法规强制组织采取这种良好的“卫生”习惯，但仍有许多机构不愿执行。

那么，只在情况变得一团糟的时候才查看日志行吗？对此最好的答案应该是“你至少应该做到这一点”，但是，除此之外，还有许多其他方法来充分利用日志内容。

下面是本书所用术语的定义。

“事件（event）是在某个环境之下发生的事情，通常涉及改变状态的尝试。事件通常包括时间、发生的事情，以及与事件或者环境相关的任何可能有助于解释或者理解事件原因及效果的细节。”（来源：http:/cee.mitre.org，具体的文章见 http://cee.mitre.org/docs/CEE_Architecture_Overview-v0.5.pdf）

事件分类根据一种或者多种分类方法为事件分组。分类方法的例子包括根据事件期间发生的情况、涉及各方、影响的设备类型等分类。

“事件字段（event field）描述了事件的某个特征。事件字段的例子包括日期、时间、源 IP 地址、用户标识以及主机标识。”（来源 :http://cee.mitre.org/docs/CEE_Architecture_Overview-v0.5.pdf)

“事件记录（event record）是事件字段的一个集合，这些字段共同描述了某个事件。与事件记录同义的术语包括“审计记录”（audit record）和“日志条目”（log entry）。（来源 :http://cee.mitre.org/docs/CEE_Architecture_Overview-v0.5.pdf)

日志（log）是一个“事件记录”的集合，“数据日志”、“活动日志”、“审计日志”、“审计跟踪”、“日志文件”以及“事件日志”等术语常常与日志的意义相同。（来源 :http://cee.mitre.org/docs/CEE_Architecture_Overview-v0.5.pdf)

审计（Audit）是指在一定环境下（例如，在一个电子系统内）评估日志的过程。审计的典型目标是评估整体状态或者识别任何值得注意或有疑问的活动。（来源 :http://cee.mitre.org/docs/CEE_Architecture_Overview-v0.5.pdf)

记录（recording）是保存与某个特定事件相关的各事件字段作为事件记录的行为。(来源 :http://cee.mitre.org/docs/CEE_Architecture_Overview-v0.5.pdf)

“日志记录（logging）是将事件记录收集到日志中的行为。日志记录的例子包括存储日志条目到某个文本日志文件，或将审计记录数据存储到二进制文件或数据库中。”(来源 :CEE Architecture Document 2011)(来　源 :http://cee.mitre.org/docs/CEE_Architecture_Overview-v0.5.pdf)

另一方面，安全事故（security incident）指的是一个或多个安全事件的发生，这些安全事件表明环境中出现了某些不良状况。这些不良状况可能是对系统的未授权访问、信息窃取、拒绝服务或者其他与某些机构相关的活动。而且，安全事故常常在清晨吵醒相关人员，因而通常很不受欢迎（其代价就是有时甚至产生“我们宁愿不知道”的错觉）。

在审计方面，日志记录可能被用在整个或部分的审计过程之中，正如我们在本书中展示的，良好的日志记录有助于审计活动的进行。但是仅有日志记录并不是审计，而且日志消息也并不一定就是审计跟踪。

应该指出，有时人们根据可信因素来区分日志记录和审计。日志记录可由系统和用户应用程序、服务器、操作系统组件产生。而审计行为由更可信的组件产生，如系统内核或可信计算（TCB）。Windows 操作系统就采用了这样的区分方式。

“警报（Alert）或警告（Alarm)”是对事件产生的响应动作，通常是为了吸引某些人（或某些东西）的注意力。有时人们说日志文件包含警报，尤其是一些入侵检测相关日志确实集合了 IDS 警报。本书中，我们将警报看作特定日志消息需要迅速向用户转发时所采取的行动。我们将把警报和警告看作是同一含义。

同样，读者们现在应该已经认识到日志是由各种不同的实体产生的，例如专用的安全设备、应用软件、操作系统以及网络基础设施组件等。我们将采用通用术语“设备”(Device)，表示安全相关日志的来源。因此，在这个定义前提下，安全 NIPS 应用程序和 Apache Web 服务器都将被作为一个设备对待。有理由相信，未来具备上网功能的家电同样会具备日志，当我们面对“面包机再次死机”的恶梦时，就可以在其日志文件中寻找解决方法。

现在我们已经掌握了这些术语，让我们通过一些例子来深入了解日志的性质及日志记录方式。

2.2　日志的概念

大多数操作系统都具备某种类型的日志记录功能，能够生成日志消息。然而，许多计算机用户（听起来令人惊讶，这其中包括系统管理员）并不知道日志的存在，更不要说去查看它们了。有三个主要原因驱使我们研究日志的来源范围。

首先，安全日志记录（Security logging）着眼于攻击、恶意软件感染、数据窃取及其他

安全问题的检测和响应。聚焦于安全的日志记录的典型例子就是用户身份认证（登录）的记录，以及意在分析某人是否具有访问某个资源的合适授权的访问决策。

其次，运营日志记录用于为系统操作人员提供有用的信息，例如通知他们系统故障和潜在的可操作条件。运营日志也可能被用于服务配给甚至是财政决策（基于访问的定价，Web服务器访问日志是日志用于业务而不仅限于IT操作的常见例子）。这一类日志来源极为广泛，涵盖了大量的日志类型。

另外，依从性日志记录通常和安全日志记录有大量重叠，因为法规通常是为了改进系统和数据的安全而制定的。知道两类依从性日志记录的区别是极其重要的：影响IT的法规和强制要求（例如PCI DSS、HIPAA、ISO、ITIL等）以及通用准则等系统法规、其他系统设计和安全强制要求。

最后，应用程序调试日志记录（application debug logging）是特殊的一类日志记录，它服务于应用程序或系统开发人员而非系统操作人员。这种日志记录通常在生产系统中被禁用，但可以在必要时启用。许多调试日志中的消息可以供完全了解应用程序或者掌握应用程序源代码的开发人员进行分析。

上述四类日志记录由几乎所有日志源（事件生产者）产生，但是可以由不同的系统（事件消费者）以不同的方式分析或消费。

操作系统产生的日志自然属于以上所讨论的范围。例如，Windows XP、Vista以及Windows 7都有一个被叫做“事件日志”（Event Log）的日志记录系统。当系统中发生特定的事件，操作系统或某个应用程序会向事件日志系统中写入一些信息。举个例子，当某Windows程序崩溃后，往往有一个查看“更多细节”的选项，大部分用户很乐意忽略它们。那些信息正是最终进入事件日志的内容。

精明的“高级用户”由于经历过Windows NT 3.5那些“糟糕的日子”，已经知道在哪找到事件查看器(Event Viewer)（见图2.1）。

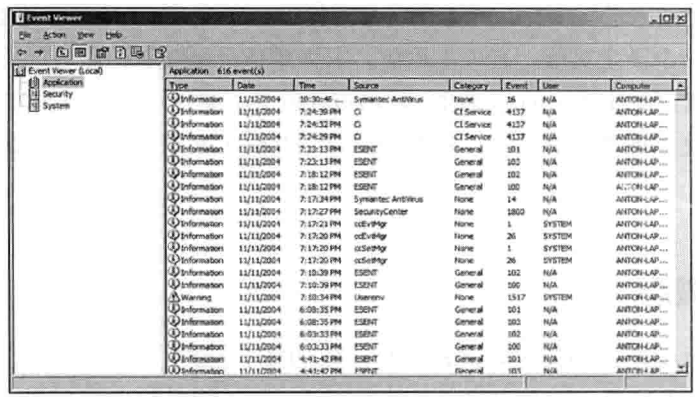


图2.1 Windows事件查看器中展示的事件日志

Unix系统的日志记录技术叫做syslog，它使人们回想起20世纪80年代早期Unix

sendmail 程序的那些日子。像 Windows 一样，操作系统本身以及某些应用程序向 syslog 中写入消息。如果系统管理员并未禁用日志记录或移除日志文件（“由于硬盘空间已满”），他可能会看到如下条消息：

```
Nov 11 22:41:00 ns1 named[765]: sysquery: findns error (NXDOMAIN) on
   ns2.example.edu?
```

这一条目包含日期、主机名以及生成此消息的服务名，还有消息自身的内容。这条特定的消息表示针对某个并不存在的域执行了一次 DNS 查询。

一些应用程序可以生成自己的日志，Web 站点管理员可能对 Web 服务日志比较熟悉。

```
10.239.46.105 - - [09/Oct/2001:11:43:34 -0400] "GET /resume/ HTTP/1.0"
   200 1100
10.222.104.178 - - [09/Oct/2001:11:48:31 -0400] "GET /security
   HTTP/1.1" 301 322
10.222.104.178 - - [09/Oct/2001:11:48:31 -0400] "GET /security/
   HTTP/1.1" 200 759
10.222.104.178 - - [09/Oct/2001:11:52:20 -0400] "GET / HTTP/1.1" 200
   1014
```

这组日志条目显示了来自不同来源（左边）的 Web 服务请求，包括请求的日期和时间、请求文档的位置以及表明文档交付状态的响应码（200，301 以及其他）。具体地说，200 表明这个文档成功显示（‘OK’代码），301 表明目标被移动到了其他位置（通过 google 搜索“http response codes”可以找到这些代码的详细信息！）。

Web 服务器日志通常会为了各种原因而被分析，包括利用率以及其他电子商务、市场类的分析。网站管理员常常挖掘 Web 服务器日志，研究用户概况，了解他们的行为、完成其他市场相关的任务。此外，Web 服务器日志还能用来研究 Web 服务器性能和网站的可用性。

下面是一个 Snort NIDS（网络入侵检测系统）“警报”日志消息的例子。

```
Nov 14 13:04:14 ns1 snort: [111:16:1] (spp_stream4) TCP CHECKSUM
   CHANGED ON RETRANSMISSION (possible fragroute) detection [TCP]
   10.15.23.130:1682 -> 10.26.50.3:25
```

注意，在这个 Snort 的例子中，这些消息被称做警报，而其他 NIDS/NIPS（网络入侵检测 / 防御系统）供应商称它为警告（alarms）或其他完全不同的名称。在所有情况下，它们实际上记录的都是一条用于向值班的操作人员或者安全分析人员告警的日志消息。

以上展示的所有日志记录系统都对安全日志分析很有用。Unix、Windows 以及 Web 服务器日志可能包含有关该组织安全态势的重要线索。

上面列出的只是常见日志记录系统的几个例子，我们将在第 9 章和第 15 章介绍这些日志系统以及其他一些系统。

2.2.1　日志格式和类型

在开始讨论格式和其他信息之前，我们应该指出，为了满足前文所述的目标，任何日志

记录机制都可以从逻辑上分为四个部分：

- 日志传输
- 日志语法与格式
- 日志事件分类学
- 日志记录的设置、配置与建议

这些主要特性将在整个报告以及本项目下生成的其他文档中使用。让我们来详细地研究它们的关键概念。

日志传输就是将日志消息从一个地方转移到其他地方的方式。

事件传输协议有许多种，例如 syslog、WS-Mangement 和许多专有的产品特定的日志传输协议，还有一些日志记录机制没有自己的传输方法（例如，只有本地日志文件）。合格的日志传输机制必须既能保证日志数据的完整性、可用性以及机密性（如果有需求的话），又能维护日志的格式和意义并使发生的所有事件都能得到正确的表示，具备正确的时间和事件顺序。日志传输中最重要的需求是保证各个日志（事件记录）以及整个事件流 / 日志的完整性[⊖]。更重要的是保存每个日志条目的正确时间戳。

以下列出一些知名的日志传输机制：

- syslog UDP
- syslog TCP
- 加密 syslog
- SOAP over HTTP
- SNMP
- 传统文件传输方式，例如 FTPS 或 SCP

例如，syslog UDP 作为迄今为止最流行的日志传输机制，应用于数以百万计的 Unix 派生系统和网络设备之上。尽管自身有多个极大的弱点（例如缺少消息送达和可用性的保证），但 syslog 依旧被认为是转移日志数据的最佳方式。

日志语法与格式定义日志消息形成、传输、存储、审核以及分析的方式。在最简单的情况下，每条事件记录可以作为文本字符串看待，事件的消费者可以在日志中执行全文搜索，期望文本具有一致性。但为了进行可靠的自动化事件分析，事件的消费者和生产者对事件记录语法的理解和认同就显得十分必要了。

下面列出一些知名的日志格式：

- W3C 扩展日志文件格式（Extended Log File Format，ELF），（http://www.w3.org/TR/WD-logfile.html）.
- Apache 访问日志 (http://httpd.apache.org/docs/current/logs.html).
- Cisco SDEE/CIDEE(http://www.cisco.com/en/US/docs/security/ips/specs/CIDEE_

⊖ 正如你注意到的，历史上流行的某些日志数据传输机制并不满足这些需求，例如 syslog UDP

Specification.htm).

- ArcSight 公共事件格式 (Common event format,CEF)(http://www.arcsight.com/solutions/solutions-cef/).
- syslog(RFC3195 和更新的 RFC5424).
- IDMEF——一种基于 XML 的格式 (http://www.ietf.org/rfc/rfc4765.txt).

对于最后一个项目，“syslog”这一术语通常指通过 UDP 协议 514 端口转移消息的方法（syslog 传输），也指以一些结构化元素创建日志消息（syslog 格式）。

特别需要说明的是，每个格式化事件记录的每个字段都包含着以某种形式表示的信息。尽管对人类来说可能显而易见，但是“sun 12 3 2010 11：11pm”和“2010-12-03T11:11:00Z”并不能清晰地表示同一个时间。而且，在第一个例子中，由于时区信息缺失，我们并不能确切地知道真正的时间，考虑到不同地区的日期表示顺序习惯不同，日期信息也存在歧义（3 月 12 日还是 12 月 3 日）。下面是日志文件格式的一些例子：

```
CEF:
CEF:0|security|threatmanager|1.0|100|detected a \| in
message|10|src=10.0.0.1 act=blocked a | dst=1.1.1.1
Apache CLF:
192.168.1.3 - - [18/Feb/2000:13:33:37 -0600] "GET / HTTP/1.0" 200 5073
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif
   HTTP/1.0" 200 2326
W3C ELF:
#Version: 1.0
#Date: 12-Jan-1996 00:00:00
#Fields: time cs-method cs-uri
00:34:23 GET /foo/bar.html
12:21:16 GET /foo/bar.html
IDMEF:
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE IDMEF-Message PUBLIC "-//IETF//DTD RFC XXXX IDMEF v1.0//EN"
"idmef-message.dtd">
<IDMEF-Message version="1.0" xmlns="urn:iana:xml:ns:idmef">
 <Alert messageid="abc123456789">
  <Analyzer analyzerid="hq-dmz-analyzer01">
   <Node category="dns">
    <location>Headquarters DMZ Network</location>
    <name>analyzer01.example.com</name>
   </Node>
  </Analyzer>
<CreateTime ntpstamp="0xbc723b45.0xef449129">
```

```
2000-03-09T10:01:25.93464-05:00
</CreateTime>
<Source ident="a1b2c3d4">
 <Node ident="a1b2c3d4-001" category="dns">
  <name>badguy.example.net</name>
  <Address ident="a1b2c3d4-002" category="ipv4-net-mask">
   <address>192.0.2.50</address>
   <netmask>255.255.255.255</netmask>
  </Address>
 </Node>
</Source>
<Target ident="d1c2b3a4">
 <Node ident="d1c2b3a4-001" category="dns">
  <Address category="ipv4-addr-hex">
   <address>0xde796f70</address>
  </Address>
 </Node>
</Target>
<Classification text="Teardrop detected">
 <Reference origin="bugtraqid">
  <name>124</name>
   <url>http://www.securityfocus.com/bid/124</url>
   </Reference>
  </Classification>
 </Alert>
 </IDMEF-Message>
```

很不幸的是，现在的大部分日志依然没能遵循任何指定或预定的格式（或者说只有消息的一小部分遵循，例如时间戳），它们仍然被认为是自由格式的文本。

日志格式有不同的特性。首先，日志文件是二进制还是 ASCII 码格式？是否可以用一个简单的文本浏览器或编辑器读取日志文件，或者需要使用转换工具以得到日志中的信息？

人类可读或 ASCII 码日志的最明显例子是 Unix syslog，下面是一个例子：

```
Nov 15 23:16:26 ns1 sshd2[12579]: connection from "10.85.179.198"
```

Web 服务器、防火墙和各种平台上的许多应用程序都采用容易查看的文本文件记录日志。

但是要注意，人类可读、文本以及 ASCII 码日志格式并非完全同义，Unicode 文本（并非 ASCII）并不少见，而且，并不是每个人都能够（或愿意）阅读嵌套关系复杂、包含冗长文本行和许多标记的 XML 格式日志文件。

二进制文件最常见的例子就是 Windows 事件日志。读者可能表示反对，认为事件日志是容易阅读的。但是阅读事件日志需要使用事件查看器。这个实用工具将二进制日志文件（一

般存储在 C:\WINDOWS\SYSTEM32\CONFIG\directory，扩展名为 Evt）转换成人类可读的形式，如图 2.2 所示。

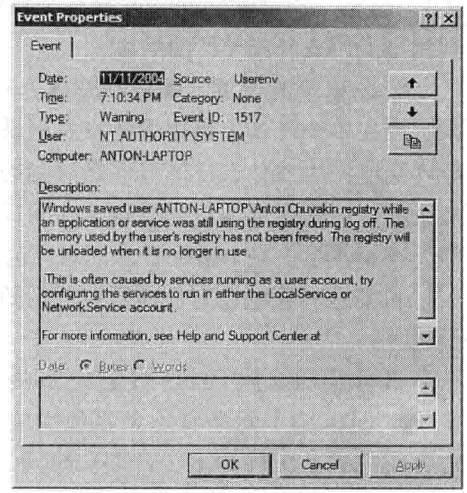


图 2.2　Windows XP 事件日志

其他的常见二进制格式还有 Unix wtmp 文件，它包括了登录记录和包含进程统计信息的 pacct。虽然严格地说不算是一种日志格式，但 Tcpdump 二进制格式可以被定义为一种网络包数据的日志记录格式，因此也可被归入这一类。

为什么一些程序员选择二进制作为日志数据格式？可以肯定的是，如果所有日志记录是基于文本、容易阅读的，对我们人类来说会更简单！但是，二进制日志记录有极具说服力的理由：性能和空间。如果某个系统每秒要收集成千上万条日志记录，即使对于现代 CPU 此任务也并不轻松。而且系统用户通常有一个合理的期望，即日志记录不会破坏系统性能。事实上，反对日志记录最常见的理由就是“它将影响性能”。而我们的回应通常是：“如果系统崩溃的时候你不知道原因，性能再好又能如何？”

二进制格式日志条目通常尺寸更小（平均信息量低），因此格式化或写入花费的处理更少。由于消息通常比 ASCII 码的消息更小，所以日志文件占据的磁盘空间更少，在传输时占用的 IO 资源也更少。

另外，解析二进制文件需要的处理通常更少，而且字段与数据类型的定义清晰，使分析更加高效。ASCII 码日志解析器不得不处理更多的数据，并且常常需要模式匹配来抽取信息中的有用内容。

压缩日志从定义上看属于二进制日志，它进一步为采用二进制日志记录提供了理由：日志记录可以进行压缩。显而易见，若日志文件是一个加密格式，则它必须采用二进制形式。采用二进制日志的其他原因包括它们更难于阅读。为什么这一点成为优势呢？因为所谓的“无名安全性（security by obscurity）”，许多人认为这使得他们的日志文件“更安全”。

另一类日志格式则是关系型数据库，它既不是真正的文本也非二进制格式，或者可以视为一种很花哨的二进制格式。关系型数据库将二进制记录存储在一个添加（称为“插入”（insertion））和读取（称为“选择”（selection））性能很高的表中，该表由数据库模式（schema）定义。数据库模式是用来定义表中的记录和整个数据库中的表的方法。

日志类型的另一个重要区分方法就是其格式是开放的还是专有的。开放格式意味着其格式在某个地方记载，可能是标准化文档（例如 ISO、ANSI 或 Internet 标准）或参考文档（例如 RFC）。专有格式可能公开也可能不公开，通常由特定的设备供应商使用。但是，专有格式通常是没有文档的，人们只有依靠供应商的日志阅读和处理工具。

理解专有文本格式的方法只能是研究它（“逆向工程”，尽管这个术语更常用于二进制格式）。但是，这种方法面临曲解某个特定字段意义、数据类型以及取值范围的风险，往往得到相当无用或者误导性的结果。

日志记录的设置、配置和建议，对于事件生产者和系统操作人员决定哪些事件应该记录，或在单个设备上启用哪些日志记录来说是最常见的途径。利用常见的事件表达方式，倡导产品应该生成什么样的事件是可能的。虽然人们期待防火墙记录日志事件，如它拦截的连接企图，但是迄今为止并没有标准的日志记录规则。同样重要的是，不仅要关注所记录的事件，还要着眼于每个事件日志记录应包含的细节信息。

目前这一领域没有任何行业范围的日志记录建议。PCI DSS 依从性是这种建议的常见例子之一。

具体而言，PCI DSS 日志记录规定（参见图 2.3）每条日志条目最少应包含用户名、事件类型、日期和时间、成功或失败的指示、受影响的组件名称以及事件的源系统（事实上这表示了一套非常有用的常见日志细节信息）。

许多其他行业组织针对日志记录的事件和详情也创造了自己的日志记录建议，本报告的下文会概括总结这些建议。

10.2 Implement automated audit trails for all system components to reconstruct the following events:	10.3 Record at least the following audit trail entries for all system components for each event:
10.2.1 All individual accesses to cardholder data	10.3.1 User identification
10.2.2 All actions taken by any individual with root or administrative privileges	10.3.2 Type of event
10.2.3 Access to all audit trails	10.3.3 Date and time
10.2.4 Invalid logical access attempts	10.3.4 Success or failure indication
10.2 5 Use of identification and authentication mechanisms	10.3.5 Origination of event
10.2.6 Initialization of the audit logs	10.3.6 Identity or name of affected data, system component, or resource.
10.2.7 Creation and deletion of system-level objects	

图 2.3　PCI DSS 需求

来源：PCI DSS 标准 v2.0

2.2.2　日志语法

任何格式的日志文件都具有语法，日志语法在概念上与语言（如英语）语法相似。人类语言语法中的句子通常包含一个主语、一个谓语、有时还包含一个表语，当然还有补语和定语。句子的语法涵盖句子成员之间的关系和他们的含义。需要注意的是，语法并不涉及消息的内容。换句话说，语法处理的是如何构造我们所要表述的内容，而不是我们使用的具体词语。

那么，在日志消息的环境下语法又有什么意义呢？当然，每条日志消息都有表征它的结构。一些类型的日志消息自身具有人类语言的部分语法，日志消息每部分由各种类型的信息模式组成。基于规则，可以定义一组通用的日志字段。而正是由于其必要性，故应存在于每条日志条目中。例如，常见的一组字段如下：

1）日期 / 时间（理想的情况下应包含时区）

2）日志条目的类型

3）产生该条目的系统

4）产生该条目的应用程序或组件

5）成功与失败的指示

6）日志消息的严重性、优先级或重要性（通常存在于所有 syslog 消息）

7）和该日志相关的任何用户活动，也可记录用户名

网络设备制造商通常在其日志中遵循上述结构，但是，缺乏某些细节的例外情况也很多。

日志的句子中有哪些内容？让我们试着对 syslog 的单行记录做次“语法分析”。作为最灵活（也最混乱）的日志格式的例子，我们将使用 syslog 的一条日志消息。

```
Nov 16 00:26:24 ns1 named[765]: check_hints: A records for J.ROOT-
   SERVERS.NET class 1 do not match hint records
```

它看上去是否很晦涩？我们故意选择了相当长且深奥的一行日志。我们可以知道“主语”（表明了“什么”）是一个命名进程，指出用户应该“check_hints”（“谓语”）指定 DNS 服务器，这个事件发生在 11 月 16 日 00:26（令人吃惊的是，syslog 在其日志时间戳中并不记载年份，这会在本章后面更详细地解释）。满足一下大家的好奇心，这里发生的事件其实就是根 DNS 服务器中的一个 IP 地址改变了，但这个 DNS 服务器（产生上面这条日志的服务器）依旧使用其配置文件中的旧信息发起查询（例如，可以参见论文《Some error message and problems with DNS》：http://www.reedmedia.net/misc/dns/errors.html）。

日志消息的语法有什么意义？日志语法对于任何一种日志数据的自动分析（本书讨论的主题）都非常重要。我们需要按照语法将日志分解成各个组成部分，才能从日志数据中得出明智的结论。

一些日志分析相关的软件产品（商用和开源）已经开发了“消息模式”（message schema）来确定各种日志类型的语法。这种模式被设计成可以满足从各种设备、系统、应用程序和其他来源日志文件产生的任何消息。

对读者来说，大部分上面的日志消息例子显然都适合于这种模式。例如如下的 Dragon NIDS 消息：

```
2004-09-11 23:01:40|dralion1|IIS:DECODE-
   BUG|10.208.231.102|10.1.1.3|2611|80|T||6|tcp,dp=80,sp=2611|
```

上例很容易融入表 2.1 所示的通用模式。这条消息表明，服务器 10.1.1.3 遭到从 10.208.231.102 发起的 IIS Web 服务器攻击（可能是由黑客或蠕虫病毒等恶意软件发起的）。此攻击与 Internet Information Services（IIS，一个标准的 Windows Web 服务）的 Unicolde 解码漏洞相关。

表 2.1　事件属性示例

字　段	示　例	字　段	示　例
EVENT	eid656539475ha1	DEVICE_TYPE	Dragon NIDS
DEVICE	dralion1	IMPORTANCE	High
EVENT_TYPE	Web Attack	DEVICE_EVENT_TYPE	IIS:DECODE-BUG
OCCUR_TIMESTAMP	2004-09-11 23:01:40	COLLECT_TIMESTAMP	2004-09-11 23:01:43

（续）

字 段	示 例	字 段	示 例
PROTOCOL_NET	TCP	PROTOCOL_APP	HTTP
SOURCE_IP	10.208.231.102	SOURCE_DNS	Evil.attacker.com
SOURCE_NBT		SOURCE_MAC	
SOURCE_PORT	2611	DESTINATION_IP	10.1.1.3
DESTINATION_DNS	Poor.Webserver.com	DESTINATION_NBT	Poor-Web
DESTINATION_MAC		DESTINATION_PORT	80
NETWORK		BYTES_XFERRED	
EVENT_COUNT	1	STATUS	

注意，在收集这些消息时还添加了一些数据（由日志分析解决方案提供）。

注意，日志分析系统添加了 EVENT 和 EVENT_TYPE 以及变量名称（NDNS，NetBIOS，以及其他），以完成 Dragon 入侵检测系统所描绘的情景。

来自“不寻常应用程序”（尽管我们不确定是否曾经见过任何“正常”的应用程序日志）等特殊日志来源的数据如何处理呢？这种情况下，人们可能会使用通用的“自定义”字段存储这类数据。虽然这样会使日志结构看上去很混乱，但是很遗憾，当前日志分析的特性就是如此。

总的来说，在做任何一种日志分析之前，对日志文件语法有一定了解都是至关重要的。有些人可以在自己头脑中执行这样的分析，但这并不影响此方法的价值，只是说明这样的分析可能很简单（也凸显了人类对机器有一定的优势）。大多数情况下，自动化日志分析系统需要理解日志语法，这种理解通常被编码成某种模板。

总体上，各种各样的系统和设备供应商会采用少数特定日志记录类型，并结合几种常见语法（在前一小节中定义）。例如，一些安全日志被记录为 XML 格式，以便进行更简单、信息更丰富的集成。很多运营日志通过 syslog 记录，采用非结构化文本或半结构化消息。与此相似，许多调试日志采用 syslog 甚至临时文本文件。而高性能日志记录常常采用二进制和专有格式。

常用设备日志记录选项在表 2.2 进行了总结。

表 2.2　各种日志机制的优缺点

	XML 日志记录	syslog 文本日志记录	文本文件日志记录	专有日志记录
消费模式	大部分机器阅读	大部分人工阅读	只适合人工阅读	只由机器阅读
常见用例	安全日志记录	运营日志和调试日志记录	调试日志记录（临时启用）	高性能日志记录
示例	Cisco IPS 安全设施	大部分路由器和交换机	大部分应用程序调试过程	Checkpoint 防火墙日志记录，数据包捕获
建议	在丰富的结构信息需要从生产系统转移到消费者手中进行分析时使用	添加 neme=value 之类的结构，简化自动分析；用于大部分运营用途	如果日志在运营中保持启用，添加结构以启用自动分析	仅用于实现超高性能
缺点	性能自相对低，日志消息尺寸大	缺少消息结构使得自动分析复杂而昂贵	通常只有开发人员可以理解	除非利用专业工具转换成文本，否则很难人工阅读

除了这些知名的日志记录方法之外，少数网络设备还会将日志记录为逗号分隔（CSV）或 ELF 格式（前文已经提到）。当然这些情况相对较少出现。

2.2.3　日志内容

至此，我们已经讨论了日志的格式和语法，但日志内容才是真正重要的。日志内容通常被定义为区分日志真正含义的分类学，因此日志事件分类学是分类所记录事件的无歧义方法。如若多个系统记录相同事件，那么可以预期对该事件的分类描述是相同的。计算机应该可以立即判定两个日志是否指向相同类型事件。为确保实现这一目标，需要一组精确定义、能够以可预见的方式组合的词语——日志分类学。想必，这些词应该可以描述活动的类型，参与的角色，结果以及其他相关的事件数据。

事件分类学现在还没有精确定义的公开标准，但是其中一些已经处于开发阶段中。但是，当今大部分安全信息、事件管理以及日志管理供应商已经在他们的产品内开发并广泛使用了自身的日志分类学方法论。不幸的是，每家供应商采用的基本分类学原则略有不同。

日志内容是一个困难的主题，有许多东西需要关注，有时不一致、不完整或充满误导。我们会从下一章开始，并在第 8、9 以及 11 章详细关注日志内容。在这里，我们给出各种信息类型的概述，指出在日志中能够期望找到什么、应该寻找什么以及在梦中才能得到的日志信息。

日志中可能包含与用户活动有关的信息：谁登录了？他们正在做什么？他们发送和接收的邮件等等。日志也能告诉某些系统出现故障或者将要出现故障，例如磁盘错误。日志还能告诉你哪些工作正在正常进行，并给出资源利用和性能的相关信息。日志还可能包括状态改变、启动和停止等等信息。日志有时能告诉你入侵尝试的相关信息，偶尔还可能会标识一次成功的入侵。

以下类型涵盖了整个安全、运营以及调试范畴的日志消息，它们是：

1）变更管理：记录系统变更、组件变更、更新、账户变更以及受到变更管理过程控制的任何其他信息；这些日志一般可分为添加、删除、更新以及修改记录。它们可能跨越安全日志和运营日志之间的分界线。

2）身份认证和授权：记录身份认证和授权决策（例如某个设备上成功或失败的登录），尤其是特权用户登录，这是最常见的安全消息，每个应用程序和网络设备都应该生成这类消息。这些安全消息在运营日志中也常常使用（例如追踪谁使用了某个特定系统）。

3）数据和系统的访问：和前几类相关，对应用程序组件和数据（例如文件或数据库表）访问的记录在安全以及性能 / 运营方面也有用途。在某些情况下，这些消息并不总是启用状态，只在敏感环境中生成。

4）威胁管理：从传统的入侵警报到违反安全策略的其他活动，这类消息由具有安全专用功能的网络设备（例如防火墙）产生。

5）性能 + 容量管理：与系统性能和容量管理相关的一大类消息，包括各种阈值，内存

和计算能力以及其他资源的利用率。这都是很常见的运营消息，不过有时在安全方面也有用途。

6）业务持续性 + 可用性管理：大部分系统在关机或开机时会记录相关日志，其他持续性和可用性日志消息与备份、冗余或者业务连续性功能利用相关。这些是很常见的运营消息，在安全方面鲜有用途。

7）杂项错误和失败：设计者认为应该吸引用户注意力的其他系统错误被划分到此类；它们不是关键运营消息，不一定需要设备管理员采取某些行动。

8）杂项调试消息：调试日志一般由个别开发人员酌情产生，很难硬性分类。大部分调试日志在运营生产环境下不启用。

2.3 良好日志记录的标准

正如这本书中各式例子所示，许多情况下日志记录的信息是不完整的，有时甚至是没用的。那么怎样才算是“良好”的日志呢？对入侵检测、资源管理或审计来说，日志消息必须包含哪些必要的信息？由于日志的种类很多，生成日志的设备更是多种多样，因此很难定义单一的标准。

通常来说，日志应该能够告诉你如下信息：

- 发生了什么（即 What，辅以适当细节信息；单纯的“某事发生”通常并不是特别有用）
- 发生于何时（即 When，若有必要，包含开始时间以及结束时间）
- 发生于何处（即 Where，在哪台主机，什么文件系统上，哪个网络接口等等）
- 谁参与其中（Who）
- 参与者来源（Where）

以上所列的是绝对必要的信息——“日志记录的 5 个 W”。许多其他领域都遵循这些准则：如新闻报道、犯罪调查等等。为了锦上添花，许多人还愿意加入下面这些信息：

- 我能从何处得到更多详细信息。
- 我怎么才能确定以上所表内容确有其事。
- 会有什么影响。

我们当然也可以梦想，期望着知道：

- 接下来会发生什么。
- 我应该再关注哪些其他事件的发生。
- 我应该怎么做。

当然，在某些情况下，上面这几项取决于特定的环境，即你的站点在某个特定事件发生后应该执行的操作与其他站点可能并不相同。

由于各种组织在一定程度上已经控制了网络设备和服务器的日志记录，应用程序日志记录显然将成为下一条战线。在习惯了整齐的 Cisco ASA 或其他防火墙日志以及 Linux 的

“password accepted”（密码被接受）消息后，安全事故的调查者不得不投身可怕的应用程序日志世界，试着去对下一波攻击做出回应。这些日志可能会在这里或那里错失细节，并且并不总是有用，但至少它们是熟悉的。

当今许多应用程序日志存在的问题简直令人难以置信：日志常常缺失，关键细节被省略，没有标准格式或内容随意出现在任何位置。在这一基础上，许多安全从业人员不得不处理伪装成安全审计日志的调试日志。

表 2.3 阐述了这两类应用程序日志的关键区别。

表 2.3　日志记录和定时问题

	审计日志	调试日志
目标消费者	安全，审计	系统操作人员、开发人员
日志记录条件	总是启用	有时启用
日志消息内容	攻击、活动、故障	故障、失效、错误
日志记录范围	预先知晓	不知晓
时间范围	数年	数小时 / 数天

与设计精良的安全审计日志相比，调试日志在应用程序框架内部的使用更频繁，由于事故响应和取证所需的关键细节在调试日志中可能没有记录，使用它们进行调查往往是令人沮丧的。第 18 章将更详细地讨论这一主题。

理想的日志记录场景

那么，如果我们可以让日志记录按照预想的方式工作，我们想看到的是什么信息，这些信息如何呈现？

我们希望看到的东西就只是那个时刻需要知道的，不多也不少。当然，这只是白日做梦，所以我们将满足于获得比这更多的信息——不同严重性的事件，以便日志分析人员根据当前情况做出决策。

从更高的层次来看，我们更愿将信息分为两类：现在就处理（紧急）的信息或后期再处理（不紧急）的信息。前者是高优先级的日志，清晰且可操作。不管某条目添加到日志中的哪个位置，都应该传达给操作人员或分析人员以便立即采取行动。当然，采取什么行动也应该很清晰。

后者包含的记录并不需要立即处理。然而当总结、时间推移或者审计 / 取证分析的时候，这些事件会提供重要的信息。

总而言之，我们乐于看到：

表 2.4 不是一个完整的列表，但是它提供了理想情况下应该被记录的事件以及记录的方式。优先级可能不只两个，但是概念上来说是相同的。事实上，真实环境下通常最少有 3 个优先级。第三个级别是无用的或不重要的“可忽略”日志消息。

表 2.4 关键日志和统计日志的对比

关键日志 / “现在行动”	统计日志 / “记在心里”
可能影响系统操作的故障	系统状态信息
成功的攻击	攻击尝试和刺探
有很大机会成功的攻击	低影响力的攻击
系统容量将要用尽或者达到某个最大值	系统接近某些参数的高值
可能导致安全和可用性问题的系统变更	各种系统变更
系统崩溃	系统启动 / 关闭
失败的登录	成功的登录
硬件失效	硬件状态信息
安全相关的配置变更	例行和自动化配置变更
检测到未授权连接	连接建立 / 终止

2.4 小结

本章为读者介绍了日志文件、审计跟踪以及其他日志相关的一般概念。

- 虽然日志具有不同的格式和来源，但是可以通过观察它们的共同特征，抽象出它们的相似性。
- 日志真实包含的信息与我们期望它包含的信息存在很大的偏差，这使得日志分析充满了挑战。

除了最基本的结论——必须记录日志！——之外，我们不得不提醒读者，日志记录的重要性只会日益增加。特别是在分布式接入以及云环境下，为了分析与安全问题相关的应用程序行为，我们最终必须控制日志记录。

软件架构师和开发人员必须掌握“日志记录”，除了网络设备和操作系统不断产生的基础架构日志之外，没有其他方法能够检测和调查应用程序级别的威胁。安全团队需要指导开发人员和架构师实施实用、高效的日志记录。

参考文献

Common event expression. Web, September 26, 2011. <http://cee.mitre.org/docs/CEE_Architecture_Overview-v0.5.pdf>.

Event log monitoring, event log management, syslog monitoring. Web, September 26, 2011. <http://www.prismmicrosys.com/EventSourceNewsletters-July10.php>.

Log analysis mailing list “World Domination” thread. <http://lists.jammed.com/loganalysis/2002/08/0012.html>.

Some error messages and problems with DNS. <http://www.reedmedia.net/misc/dns/errors.html>).

Syntax resources on the web. <http://www.utexas.edu/courses/linguistics/resources/syntax/>.

第 3 章 Chapter 3

日志数据来源

本章要点

- 日志记录来源
- 日志来源分类

3.1 概述

日志分析过程的第一步是了解日志数据的来源。有些应用程序和系统默认记录日志，有些则不记录。在某些情况下，应用程序可能发送日志数据，但是没有被接收。在本章中，我们将展示如何启用系统和应用程序的基本日志记录。因为有些应用程序 / 系统的默认设置不一定充足，我们还要说明如何调整应用程序设置，改进日志的质量。在少数情况下，我们展示一些工具，可以利用应用程序没有提供的信息扩增日志。总体的目标是提供示例，帮助你考虑所在环境中如何设置日志记录，以及对现有日志的扩增。

3.2 日志来源

我们首先来看看各种日志记录系统及它们收集的数据格式。日志来源分为两类：

- 基于“推”
- 基于“拉”

配置日志数据源

本章将提供 Unix 的 syslog 配置示例，而不提供不同类型的日志来源示例，这是因为大部分日志来源都有供应商提供的丰富文档，如果是开放源码的日志来源，可以用 Google 搜索提供工具配置的指南。

对于基于“推”的日志来源，设备或者应用程序向本地磁盘或者通过网络发出消息。如果通过网络，你必须配备一个日志收集器来接收消息。3 种主要的基于“推”来源是 syslog、SNMP 和 Windows 事件日志。日志消息通过这些协议传输。从技术上说，Windows 事件日志包含协议、传输机制、存储和读取。

对于基于“拉”的日志来源，应用程序从来源拉取日志消息。这种方法几乎总是依赖客户 - 服务器模型。以这种方式运行的系统通常以专有格式保存日志数据。例如，Checkpoint 提供 OPSEC C 程序库，开发人员可以用它编写应用程序，拉取 Checkpoint 防火墙日志。其他产品使用 MSSQL、Oracle、MySQL 等数据库存储数据。从数据库拉取日志稍微容易一些，可以用脚本或者程序完成。

我们来看看三个最常见的日志来源协议。首先是 syslog 协议。

3.2.1 syslog

在第 1 章和第 2 章中你已经看到了 syslog 消息示例。本节更多的是关于协议本身的。syslog 被 Unix 内核和许多应用程序用来记录日志消息，它最初是用于收集调试信息的。因此，它对于安全日志分析有一些限制，不是最优的。尽管如此，syslog 已经成为了基于 UNIX 的系统中记录应用程序事件的最常用方法。

syslog 包含 syslog 守护进程（syslogd）。它通常分别在开机和关机时启动和停止。应用程序通过 syslog(3) 库调用与 syslogd 通信。syslogd 通过 Unix 域套接字从应用程序和内核接收日志记录 syslogd 还可以选择通过端口 514 上的 UDP 消息，从远程主机接收数据。现代版本和 syslog 替代品（如 rsyslog 和 syslog-ng）还能使用 TCP。

syslogd 的行为由配置文件控制，这个文件通常是 /etc/syslog.conf。守护进程从 Unix 域套接字（domain socket）/dev/log（在不同的 Unix/Linux 变种中，该名称可能不同）读取日志消息，并写入一个或者多个输出文件，或者通过 UDP 将日志转发到一台收集主机。对配置文件的更改需要发送一个 SIGHUP 到 syslogd 进程，重新读取新配置。

syslog 通常在系统自举时启动，但是可以运行“syslogd”手工启动。syslogd 的标准参数可以指定配置文件的位置、“标记时间间隔”（下文将作解释）、是否接收来自网络的数据以及 Unix 域套接字的路径。选项的特殊名称在不同的 Unix/Linux 变种中各有不同。例如，Linux 使用“-r”启用从主机接收日志（“remote”），而 OSX 使用“-u”。有些版本的 syslogd 自动接受远程消息，不需要任何命令行参数。这些选项通常有默认值，命令行参数用

于覆盖默认值。特定于操作系统的信息可以查询本地的 man 页面。

默认 syslogd 配置文件（/etc/syslog.conf）的内容在不同发布版本中也不一样。大部分配置默认情况下不记录所有日志，有些对不同的程序记录不同的日志信息。你应该总是检查 syslog 配置文件，以确定日志中是否记录了你需要的信息。

1. 用 syslogd 记录基本日志

那么，如何用 syslogd 中开始日志记录？我们建议采用一种简单的方法：将所有日志记录到本地主机上的一个文件中（我们将在后面讨论长期记录所有信息是不是好主意）。将所有日志记录到单一文件有两个好处。首先，你可以浏览一个文件查看所有日志信息。如果你不知道记录了什么，那么如何做出应该把“什么”记录到“哪里”的决策？其次，因为后面我们将把这些消息转发到一个集中日志主机，本地拷贝保留下来用于故障排除和备份用途。

要在一个文件中记录所有日志，可以在 syslog.conf 中加入如下文本（含义和语法将在后面解释）：

```
*.debug     /var/log/messages
```

如果你喜欢，可以使用不同的路径名，这里使用的只是为了举例。如果选择不同路径名，你可能必须创建这个（空）文件，然后 syslog 才能写入它（我们将在本章后面更多地讨论它）。

你可以注释掉文件中的其他行，避免多余地将相同的信息写到一个以上的文件中。在你想要注释或者禁用的行前面添加一个“#”。

用 ps 获取 syslog 守护进程（syslogd）的进程 ID：

```
# ps -ef | grep syslog
root  3344  1 0 Oct20 ?        00:00:01 syslogd -m 0
```

向该进程发送一个 SIGHUP：

```
kill -HUP 3344
```

查看文件 /var/log/messages，你应该看到如下消息：

```
Oct 23 17:08:22 somehost syslogd 1.4.1: restart.
```

这条消息后面可能还跟着许多其他消息。这就对啦！你现在正在记录发送到 syslog 守护进程的所有信息。值得注意的是，在某些 Unix 版本上，syslog 守护进程默认不监听来自远程系统的入站请求。这意味着，具备可发送 syslog 日志消息的应用程序的远程系统无法在中心位置（也就是集中日志服务器）记录日志。在大部分情况下，需要在启动时向 syslog 守护进程传递一个特殊标志。想起来了吧，在大部分 Linux 上是“-r”选项，而 OS-X 使用 -u 选项。查询你的 syslog 守护进程手册确定该选项。

很快，我们将讨论日志文件中看到的消息的含义。但是现在，我们将从配置的角度解释

你刚刚进行的操作。

2. syslog 消息分类

syslog 消息有两个属性供 syslogd 使用，以决定如何路由它们：设施和优先级。设施用于指定消息生成位置的一般分类。

设施是一个固定集合中的一项，你不能制作自己的设施并开始使用它。你可以使用的设施集合包括：

auth	非敏感身份认证信息。在 Linux 上，也可以使用名称“security”
Authpriv	敏感身份认证信息（你可能不希望出现在可读的日志文件中的内容）
Cron	Cron 活动
Daemon	来自任何守护进程的消息
Kern	内核生成的消息
Lpr	来自打印系统
Mail	来自邮件传输代理的消息
Mark	在固定间隔生成的内部消息
News	来自 Usenet 新闻系统的消息
syslog	来自 syslog 守护进程的消息
User	来自命令行工具的消息（例如 logger）
Uucp	来自 uucp 系统的消息
local0 through local7	来自”定制“应用程序的消息。用于内部应用程序等

这些设施的名称在不同的 Unix/Linux 发布版本中稍有不同，有些操作系统有少数其他系统没有的分类。例如，OS-X 有一个名为“ install”的设施。Solaris 9 没有“ syslog”或者“authpriv”设施。你可以查询操作系统的 man 页面。

注意，由应用程序程序员选择设施，因此对这一选择没有任何限制。即使程序不是邮件程序或者内核程序，程序员也可以选择使用 Mail 设施或者 Kern 设施。

3.“Mark”设施

“ Mark”是一个特殊设施，这个设施由 syslog 内部使用，在固定间隔内生成一个标记消息，如：

```
Oct 23 17:54:00 somehost -- MARK -
```

标记设施的目的是验证 syslog 守护进程正在工作，即使没有接收到任何消息。标记消息对于监控系统是否正常运行以及确定主机何时崩溃很有用，你可以使用崩溃前的最后一个标记消息的时间来（粗略）确定故障时间。

如果在命令行上没有指定，默认的时间间隔是 20 分钟，但是有些发布版本的启动脚本完全关闭标记消息（将消息时间间隔设置为 0）。

4. syslog 优先级

消息的优先级表示消息的重要性。

可用的优先级如下（按照敏感性递增的顺序）：

- debug.
- info.
- notice.
- warn.
- err.
- crit.
- alert.
- emerg.

同样，与指定事件相关的优先级也是由应用程序的程序员选择的，所以不能保证它真正意味着严重性。

设施和优先级的实际应用是 syslog 守护进程用它来形成消息过滤的基本形式。这种过滤在 syslog.conf 文件中规定。

5. syslog.conf

syslog.conf 文件包含如下格式的一行或者多行：

<selector>... <tab> ... <action>

<selector> 指定所在行适用的消息类型，<action> 指定这条消息与什么操作有关。

<selector>（选择器）是设施和优先级的组合，以句点分隔，如 daemon.debug。选择器将匹配指定优先级或者更高优先级的该设施的所有消息。这意味着 kern.info 匹配从 kern.info 开始到 kern.critf（或者最高优先级）的所有消息。传统的 syslog 不允许指定单个优先级匹配，只能指定最低优先级匹配通过逗号分隔，可以为单个优先级指定多个设施，例如，daemon,lpr.debug 指定 daemon.debug 和 lpr.debug。“ * ”可以用于代替设施名，指定所有设施。多个选择器可以用“ ; ”分隔，放在同一行里，例如 daemon,info;mail.crit。最后，“ none ”关键字可以作为优先级，它将取消同一行上的任何选择器，这可以用于和通配符配合，选择除了特定设施以外的所有设施，例如“ *.info;mail.none ”将选择 info 优先级上除了邮件设施之外的所有设施的消息。

选择器和操作（action）由 <tab> 字符分隔。虽然新版本的 syslogd 除了 <tab> 外还接受 <space>，但是如果在一行上出现 <space>，旧版本的 syslogd 将会崩溃。

对于选择器所采取的操作可以是如下三种之一：将消息附加到一个文件或者管道、转发消息到另一台主机上的 syslogd 以及将消息写入用户终端。

最常用的操作是指定文件的完整路径，将消息附加到一个文件。例如：

```
mail.debug  /var/log/mail.log
```

在许多 syslogd 版本上，该文件必须已经存在，syslogd 才能写入它；该文件如果不存在，syslogd 不会创建它。如果文件不存在，你可以用 touch< 文件名 > 或者 cp/dev/null< 文件名 > 来创建文件（使用 touch 的好处是如果文件已经存在，该文件不会被损坏，这对于 shell 脚本很有用）。

可以使用同样的操作，将消息写入设备。最常见的例子是写入系统控制台（/dev/console）。在远古时代，控制台实际上就是硬拷贝终端（例如 decwriter），“重要”消息应该被写入控制台，以便在系统崩溃时，管理员能够查看打印输出，并尝试推断崩溃时发生的状况。当然，在那个时代，系统比现在慢得多，运行的设备也较少，所以打印到控制台的信息量是可以控制的。而且，在那个时代，磁盘很昂贵，而纸张则相对便宜。

将日志消息转发到另一台主机通过在主机名前加上“@”作为操作来完成，例如：

```
*.info   @loghost.mydomain
```

日志主机可以是一个主机名，也可以是 IP 地址。指定 IP 地址的好处是 syslog 守护进程启动时，消息在 DNS 不能正常工作的情况下也可以转发。消息通过 UDP 发送给远程主机，不期待任何响应，所以远程主机的故障也不会使 syslogd 无法在本地主机上运行（这既有好处又有坏处，我们将在下面讨论）。

最后，第三种可能的操作是将消息写入用户终端。当然，用户必须登录，才能显示消息。这通过在操作中指定用户名完成，例如：

```
kern.crit               kjs
```

这个例子将导致 crit 或更高优先级的所有内核消息写入 anton 所拥有的所有 tty。注意，这一操作应该审慎使用，向用户“anton”发送 *.debug 可能使 anton 非常不愉快。消息可以通过指定“*”为操作，发送给所有用户。这种特殊情况通常保留用于非常关键的事件，一般表示即将发生的系统崩溃，如：

```
kern.emerg      *
```

有些 syslog 守护进程有额外的功能，可以通过配置文件控制。例如，有些 syslogd 允许你限制日志主机可以从哪些主机接受 syslog 消息。我们建议你阅读系统的 Fine Man 页面，找出可能拥有的额外功能。

6. 标记时间间隔

如上所述，“标记”消息的频率通常可以设置为命令行参数，粒度为一分钟。默认值为 20 分钟。确定标记消息的“正确”时间间隔需要折中考虑。小的值（如 1 分钟）将更精确地确定主机（至少 syslog 守护进程）启动或者停机的时间。但是，如果你有数千台主机每分钟都向集中日志主机发送标记消息，你的网络上就会有巨大的相关流量。但是，你可能希望指

定某个标记时间间隔，以便验证 syslog 真的在工作。

7. syslogd 输出

syslogd 写入 ASCII 文本消息，以换行结束。所以阅读该文件不需要特殊的查看器，任何文本查看器都可以使用。不过，我们不建议使用“ vi”或者其他编辑器检查日志文件。首先，日志文件可能非常大，你的编辑器可能被堵塞。更重要的是，用文本编辑器阅读日志文件造成不当修改的风险，如果你需要将日志数据用于法律或者商务用途，这是不能接受的。除非你打算修改文件，不在文件上使用文本编辑器是系统管理的好习惯。

8. syslog 协议

多年以来，syslog 协议的标准 RFC 是 RFC3194（http://www.ietf.org/rfc/rfc3164.txt）。现在，RFC5424（http://tools.ietf.org/search/rfc5424）是新提出的 syslog 协议标准草案。换言之，RFC5424 将淘汰 RFC3194。

RFC5424 是旧的 syslog 协议的必要更新。协议中最大的变化之一是遵循 RFC3339(http://tools.ietf.org/search/rfc3339）的时间戳规范。旧协议对时间戳没有太多的规定。如果你接收的日志消息包含月、日、时和秒信息，就算运气了。时间戳中通常没有年份信息和时区信息。从分析的角度来说，这造成了很大的困难。RFC5424 还在 syslog 中添加了 name=value 等结构化数据，显著地简化了自动化日志分析。

我们鼓励你查阅 RFC，对协议的概念和功能有基本的感性认识。

3.2.2　SNMP

SNMP 设计用于满足网络管理员不断增长的需求。从 20 世纪 90 年代初起，SNMP 已经集成到几乎所有你能想到的网络系统中，包括许多网络安全系统。SNMP 是查询和配置设备的一种协议。SNMP 陷阱和通知是设备在特定事件发生时生成的特殊 SNMP 消息。虽然 SNMP 协议整体来说不是一个日志记录系统，但是 SNMP 陷阱和通知可以看作日志消息的类型。虽然许多网络设备能够通过 syslog 发送事件信息，但是有些设备不能，特别是旧设备，因此 SNMP 陷阱和通知是从设备获得其他途径不能收集的事件信息的一种方法。在某些情况下，通过 SNMP 发送的信息类型与通过 syslog 发送的不同。

SNMP 有多个版本，通常称作 SNMPv1、SNMPv2 和 SNMPv3。详细介绍 SNMP 超出了本书的范围，所以我们只关注陷阱 / 通知、获取和设置。下面的小节介绍这些主题，但是首先我们必须讨论管理器和代理的概念。

1. 管理器和代理

SNMP 管理的设备通常由网络管理站（NMS）控制。NMS 定期轮询设备，查询状态信息，在必要时发送配置更改。NMS 还监听陷阱或者通知。利用这种方式，NMS 的功能类似

于集中日志收集器。支持 SNMP 的主要理由是日志事件可以导出到传统的 NMS，如 HP 的 OpenView。但是它们也可以导出到日志收集系统。

配置 SNMP 陷阱的具体方法对于每种设备各不相同。在如下链接的文档更详细地描述了如何在 IOS12 上配置 SNMP 的所有特征：http://www.cisco.com/en/US/docs/ios/12_2/configfun/configuration/guide/fcf014.html。

这个文档很好地结合了 SNMP 的一般介绍和配置 SNMP 的实用指南，虽然针对的是 IOS，但是十分通用，可以用作其他系统的指南。

和 syslog 一样，必须有某个设备监听 SNMP 陷阱。接收者可以是 NMS，你也可以在喜欢的 Unix 版本上运行某些 SNMP 守护进程。Net-SNMP 是最流行的开源 SNMP 工具箱。它由使用的命令行工具和一个 SNMP 陷阱守护进程（snmpd）组成，可以运行于大部分 Unix 和 Windows 上。Net-SNMP 的官方网站是 http://www.net-snmp.org/。

2. SNMP 陷阱和通知

SNMP 陷阱是 SNMPv1 协议的一部分。SNMP 通知是 SNMPv2 和 SNMPv3 协议的一部分。陷阱和通知之间的关键区别是通知包含了接收者向发送者发回确认的功能。设备必须经过配置才能产生消息，包括什么事件生成消息，向哪里发送消息。

和 syslog 一样，SNMP 通过 UDP 实现，因此和 syslog 有同样的可靠性问题。2002 年，CERT 发现 SNMP 的许多商业实现有漏洞。今天，这个问题大体上已经被人遗忘，供应商试图进一步解决这些安全问题。但是，对这些问题的存在必须引起注意。SNMPv2 通知允许从接收者发回确认，至少提供了比单向 UDP 消息更可靠的消息传递。

SNMPv1 陷阱使用明文传送，且没有身份认证，这也和 syslog 一样。因此，它们也容易遭到相同类型的欺骗攻击。SNMPv2 通知也以明文发送。SNMPv3 有可选择的消息认证，可以抵御欺骗攻击，代价是占用了发送者和接收者上的一些 CPU 周期。

3. SNMP get

SNMP 协议允许“get”（获取）操作，你可以用它从一个设备或者系统上读取信息。你能够得到什么？这很大程度上取决于设备的实现。例如，路由器将跟踪每个接口的发送和接收字节数等信息。操作系统通常允许你获得 CPU 使用、内存使用等信息。从设备或者系统中可以获得有助于日志分析的信息。但是在实践中，这既不实用、也不现实。在网络管理界有个概念叫做陷阱导向轮询（trap-directed polling）。这意味着，当你接收到某种 SNMP 陷阱，使用 SNMP get 轮询设备以读取附加信息，进一步关联或者验证刚刚收到的陷阱。但是一般来说，发送陷阱的安全设备所作的只是：发送陷阱，但是并不跟踪任何可以“获取”的信息。

4. SNMP set

为了完整性，我们将简短地讨论 SNMP set（设置）。顾名思义，SNMP set 允许你更改

远程系统上的某个数值。例如，网桥实现某种规范，允许你使用 SNMP set 开关交换机端口。为此，你必须知道连接到特定端口的主机的 MAC 地址。

5. SNMP 作为日志数据替代方案的问题

如前所述，syslog 消息的一个大问题是没有标准的格式。这意味着，供应商 ABC 的日志消息很可能与供应商 XYZ 的不同。然而，对于 SNMP，情况也只是略好一些。SNMP 使用了所谓的 MIB（Management Information Base，管理信息库）。概略地说，这是特定系统所支持的陷阱和通知的定义。下面是来自 Snort MIB 的一条通知：

```
sidaAlertGeneric NOTIFICATION-TYPE
   OBJECTS { sidaSensorVersion,
      sidaSensorAddressType, sidaSensorAddress,
      sidaAlertTimeStamp, sidaAlertActionsTaken,
      sidaAlertMsg,
      sidaAlertMoreInfo, sidaAlertSrcAddressType,
      sidaAlertSrcAddress, sidaAlertDstAddressType,
      sidaAlertDstAddress, sidaAlertSrcPort,
      sidaAlertDstPort, sidaAlertImpact,
      sidaAlertEventPriority, sidaAlertSrcMacAddress,
      sidaAlertDstMacAddress }
   STATUS current
   DESCRIPTION
      "The Sida Alert Generic Trap is sent whenever an
      event is detected by snort (rules) and no specific
      Alert is found applicable."
   ::= { sidaAlertTypes 1 }
```

查看这一通知的结构细节和含义并不重要。我们只想对 SNMP 有一个总体的概念。需要指出的一点是定义中的 OBJECTS 部分。这是将在通知中发送的单独事件细节。例如，sidaSensorVersion 在同一个 MIB 中的定义如下：

```
sidaSensorVersion OBJECT-TYPE
   SYNTAX SnmpAdminString
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
   " the version number of the sensor that detected the event."
   ::= { sidaSensorEntry 3}
```

这是我们了解了特定细节的样子，在这个例子中，它是一个字符串，表示报告该通知的 Snort 版本。

遗憾的是，尽管 MIB 为良好格式、容易理解的消息提供了一个强大的框架，但是供应商很少提供这样的 MIB。供应商用各种办法歪曲它。例如，MIB 可能与设备或者系统实际发送的陷阱或者通知不符。或者，对底层 SNMP 实现进行了更改，但是 MIB 没有更新，很快就变得不同步。然后，当我们这些最终用户使用 MIB 来理解陷阱时，即使没有完全被误导，也至少会被弄糊涂。

最终，供应商没有花时间创建和上面的 Snort 示例类似的 MIB，而是创建一个发送单变量的陷阱，这个变量就是一个自由格式的文本字符串。结果是，SNMP 陷阱和 syslog 消息一样难以解析。实际上，许多供应商使用的是和 syslog 消息相同的消息，并将其包装在 SNMP 陷阱中。

所以，SNMP 可能不是收集日志信息的最佳方式，但是它可能是从某些设备获取信息的唯一手段。

3.2.3 Windows 事件日志

Microsoft 在很久以前就决定要创造自己的日志生成和收集系统。这一系统被称作事件日志（Event Log）。该系统经过了多年的发展，几乎和 Windows 出现得一样早。现在的事件日志有一些高级特性。事件日志主要用于收集和查看两类日志：

- Windows 日志
- 应用程序日志

Windows 日志至少包括应用程序、安全和系统。最重要的是安全日志。这是登录、注销、资源访问（共享、文件等）记录的地方。应用程序日志不言自明。应用程序可以写入该日志，传递状态、错误和其他值得记录的项目。

我们来简单地看一下事件日志。你可以这样启动查看器：控制面板→系统和安全→管理工具→查看事件日志。这将打开事件查看器。图 3.1 展示了查看器。

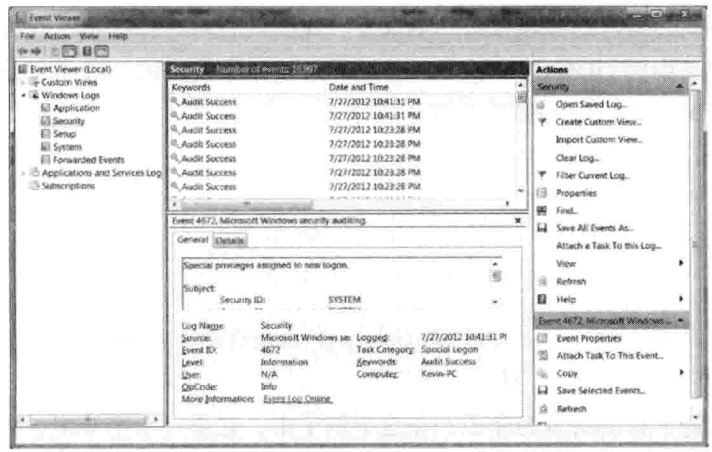


图 3.1　Windows 事件查看器

在窗口的左侧窗格是各种日志消息类型。中间是你所选择的日志消息（例中是安全日志）。在中间的日志消息下方是日志消息的细节。如果双击日志消息，将得到类似图 3.2 的弹出式消息。

图 3.2 展示了你所选择的日志消息细节。根据日志消息类型，你将看到如下内容：

事件 ID（图 3.2 中是 4624）。

账户名称

域

资源（文件或者目录访问时）

状态（请求成功与否）

内建的日志查看器所能显示的内容已经很好了，但是你可以使用 http://eventid.net/ 更深入地了解事件的含义。如果你有事件 ID，可以用这个网站获取该事件的更多细节。图 3.3 是该网页关于事件 ID4624 的片断。

你可以看到，这里的内容比图 3.2 中实际日志的细节更多。

图 3.2　事件 4624 的细节

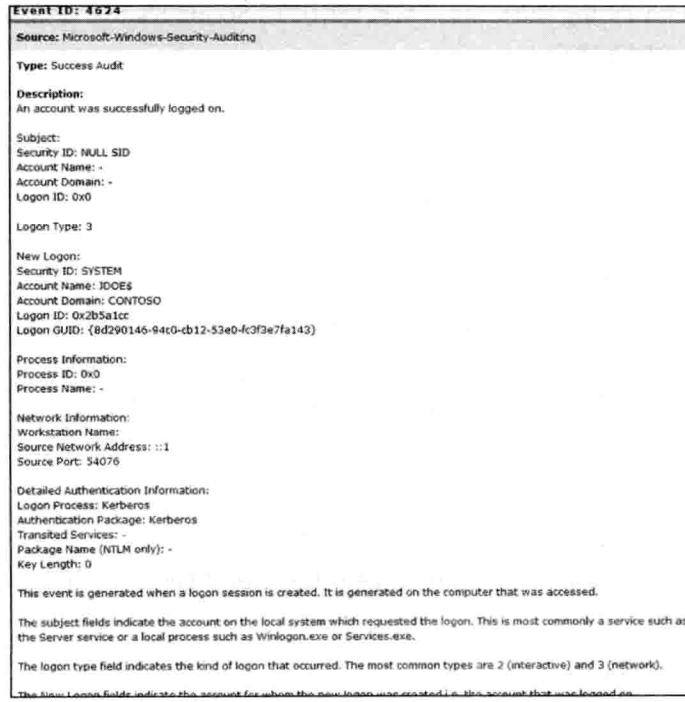


图 3.3　Windows 事件查看器

第 8 章和第 15 章讨论了 Windows 事件查看器这种宝贵资源的易管理性。

3.3 日志来源分类

本节提供了一些生成日志数据的系统和应用程序实例。

3.3.1 安全相关主机日志

这一类别涵盖由操作系统组件、各种网络服务日志和其他运行于系统之上的应用程序生成的主机日志。虽然许多消息只是（或者主要是）为性能跟踪、审计或者故障排除而生成的，但是大部分在安全上都有作用。

1. 操作系统日志

操作系统记录各种消息。我们来研究一些操作系统生成的安全相关消息：

- 认证：用户已经登录、无法登录等。

  示例（Linux syslog）：

  ```
  Jan 2 08:44:54 ns1 sshd2[23661]: User anton, coming from 65.211.15.100,
     authenticated.
  ```

  这个例子是 Linux syslog 中的一行，与远程用户用 Secure Shell 守护进程认证相关：

- 系统启动、关闭和重启。

  示例（Linux syslog）：

  ```
  Nov 4 00:34:08 localhost shutdown: shutting down for system reboot
  ```

  这个例子中的 Linux syslog 与系统关闭相关。

- 服务启动、关闭和状态变化。

  示例（Solaris syslog）：

  ```
  Nov 5 13:13:24 solinst sendmail[412]: [ID 702911 mail.info] starting
     daemon (8.11.6+Sun): SMTP+queueing@00:15:00
  ```

  这个例子是与 sendmail 守护进程启动相关的 Linux syslog 行。

- 服务崩溃。

  示例（Linux syslog）：

  ```
  Jan 3 12:20:28 ns1 ftpd: service shut down
  ```

  这个例子是与 FTP 服务器偶然关闭（可能因为崩溃或者 kill 命令）相关的 Linux syslog 行。

- 杂项状态消息。

  示例（Linux syslog）：

  ```
  Nov 20 15:45:59 localhost ntpd[1002]: precision = 24 usec
  ```

这个例子是与时间同步守护进程（NTPD）相关的 Linux syslog 行。

总体来说，操作系统消息被视为安全相关有两个主要原因：

1）它们可用于入侵检测，因为成功和失败的攻击通常在日志中留下独特的痕迹。大部分入侵检测系统（HIDS）和安全信息及事件管理系统 (SIEM) 收集这类消息，做出过去和将要出现的威胁的判断（在日志中发现攻击者侦察活动的痕迹时）。

2）它们对事故响应（见第 16 章）也很有用，因为尽管有各种安全防护措施，成功的攻击仍然可能发生。正如我们在许多章节中所提到的，日志在事故响应中是至关重要的，因为它们使调查者能够“组合”入侵拼图中互相脱节的各个部分。

2. 网络守护进程日志

网络守护进程通常记录如下类型的安全相关消息：

- 建立到服务的连接。

  示例（Linux syslog）：

  ```
  Dec 26 06:45:14 ns1 popper[14251]: (v4.0.5) POP login by user "anton"
     at (10.192.17.92) 10.192.17.92
  ```

  来自 Linux syslog 的这条消息显示了远程用户“anton”成功地连接到 POP3 邮件守护进程。

- 失败的服务器连接。

  示例（Linux syslog）：

  ```
  Dec 28 01:54:16 ns1 xinetd[14923]: FAIL: telnet libwrap
     from=210.93.83.28
  ```

  来自 Linux syslog 的这条消息说明 telnet 服务的一次失败连接（因为访问控制）。

- 连接建立，但是不允许访问。

  示例（Linux syslog）：

  ```
  Dec 13 08:45:00 ns1 sshd2[18120]: connection lost: 'Connection closed.'
  ```

  来自 Linux syslog 的这条消息说明对 Secure Shell 服务器的连接不成功。

- 各种故障消息。

  示例（Linux syslog）：

  ```
  Dec 26 06:47:12 ns1 sendmail[14259]: iBQBkZc14259: lost input channel
     from [10.8.206.4] to MTA after rcpt
  ```

  来自 Linux syslog 的这条消息说明 sendmail 守护进程无法继续与客户端通信（可能是一个垃圾程序）。

- 各种状态消息。

  示例（Linux syslog）：

  ```
  Dec 26 06:47:12 ns1 sendmail[14259]: iBQBkZc14259: from=<cqywejwywwno@
     fghjgh.com>, size=0, class=0, nrcpts=2, proto=SMTP, daemon=MTA,
     relay=[10.10.206.4]
  ```

来自 Linux syslog 的这条消息表示一次成功的电子邮件传输。

网络守护进程日志通常和一般的操作系统日志一样有用。实际上，它们常常被记录在同一个位置，例如，在 Unix 和 Windows 上，它们使用相同的日志记录机制。

网络守护进程提供了远程进入系统的最常见途径，许多攻击以此为目标。因此，建立健全的日志记录对于这种环境很关键。

3. 应用程序日志

应用程序还记录各种有趣的消息。我们可以将应用程序记录的日志归纳为如下列表：

- 应用程序用户活动
- 特权用户活动
- 关键的例行活动
- 重新配置

3.3.2 安全相关的网络日志

这一类别涵盖了网络基础设施生成的网络日志。路由器和交换机生成各种有趣的日志消息，这与它们的操作以及通过的流量相关。

网络基础设施日志

网络基础设施包括路由器、交换机和其他组成网络、将桌面和服务器绑定在一起的设备。来自这类设备的日志在安全中起着关键的作用。

最常见的消息包括如下类别：

- 登录和注销
- 建立服务连接
- 出站和入站传输字节数
- 重新启动
- 配置更改

3.3.3 安全主机日志

这一类别包括来自在主机上运行具备安全保护功能的应用程序的主机日志。和上面提到的和安全有关或无关的日志不同，这些安全日志总是有趣的，因为它们与攻击、入侵、感染等相关。但是，在许多情况下，安全机制可能会撒谎，造成各种类型的虚假警报（例如著名的“假阳性”）。

主机入侵检测与预防

从 20 世纪 90 年代初第一代商业化系统出现以来，主机入侵检测系统（HIDS）和入侵预

防系统（HIPS）的定义和任务已经得到了发展。顺便提一句，最早的系统实际上查看日志，试图对这些日志应用入侵跟踪特征。HIDS 的功能已经被扩展，还监控文件系统的变化和其他未授权的系统修改。

现在，这类系统能够检测和拦截各种网络、操作系统和应用程序攻击。HIDS 只发出警报，HIPS 还可以根据特征、动态规则和其他机制拦截攻击。

这类系统生成的大部分事件记录与如下方面相关：

- 检测到的侦察或者探查行为。
- 对可执行文件的修改。

  示例（Dragon HIDS）：

  ```
  2002-10-11|10:38:38|labdragon-hids|FTP:NESSUS-
  PROBE|0.0.0.0|146.127.94.13|0|0|I||0|target:146.127.94.13,file:messages|
  ```

  来自 Dragon 主机传感器（以前称为 Dragon Squire）的这条消息说明，通过监控 FTP 日志，发现了一次 Nessus 漏洞扫描器探查。
- 检测到攻击

  示例（Linux syslog）：

  ```
  Dec 26 06:47:12 ns1 sendmail[14259]: iBQBkZc14259: lost input channel
     from [10.8.206.4] to MTA after rcpt
  ```

  来自 Linux syslog 的这条消息说明：
- 检测并拦截攻击。

  示例（Linux syslog）：

  ```
  Dec 26 06:47:12 ns1 sendmail[14259]: iBQBkZc14259: lost input channel
     from [10.8.206.4] to MTA after rcpt
  ```

  来自 Linux syslog 的这条消息说明：
- 检测到成功的入侵。

  示例（Linux syslog）：

  ```
  Dec 26 06:47:12 ns1 sendmail[14259]: iBQBkZc14259: lost input channel
     from [10.8.206.4] to MTA after rcpt
  ```

  来自 Linux syslog 的这条消息说明：
- 不安全的系统重新配置或者损坏。

  示例（Dragon HIDS）：

  ```
  2002-10-11|10:32:11|labdragon-
  hids|FILE:DELETED|0.0.0.0|146.127.94.13|0|0|I||0|target:146.127.94.13,f
  ile:/etc/inetd.conf|
  ```

  来自 Dragon 主机传感器（以前称为 Dragon Squire）的消息显示了关键系统文件删除警报。

- 身份认证或者授权失败。

  示例（Dragon HIDS）：

```
2002-10-11|10:38:38|labdragon-hids|LOGIN-
FAILED|0.0.0.0|146.127.94.13|0|0|I||0|target:146.127.94.13,file:messag
es|
```

3.4 小结

本章介绍了许多基础知识，重点是用例子介绍概念。本章中提供的思路并不全面。它们只是对可能发生的情况做一些基本介绍，使你能在自己的环境中解决问题。

第 4 章 Chapter 4

日志存储技术

本章要点

- 日志留存策略
- 日志存储格式
- 数据库存储日志数据
- Hadoop 日志存储
- 云和 Hadoop
- 日志数据检索和存档

4.1 概述

日志的存储和快速检索分析是组织中的一个关键问题。中小企业留存的日志记录会很快地增长到 TB 或 PB 级。这些数据在组织中以各种不同的格式存储，我们会探讨众多的留存策略，以期匹配环境和依从性的需要。

4.2 日志留存策略

开发日志留存策略，将为本章所介绍的留存主题打下坚实的基础。你为自己组织设计的策略会推动许多决策，包括你的日志数据的存储类型、大小、成本、检索速度以及存档和销毁需求。

我们将根据下面说明的因素，经历一个创建留存策略的过程。这个计划需要来自组织中安全、依从性和业务管理部门的利益相关方的参与，才能创建一个合乎逻辑、实用并具备合适范围的日志留存计划，它不会有过高的成本，但仍可满足组织需要。开发自己的日志留存策略时，应该认真审视以下各项：

1）评估适用的依从性需求：

在当今的许多行业中，诞生了一大批健全的依从性需求。这方面的例子包括支付卡行业数据安全标准（PCI DSS），它规定了非常具体的日志留存周期——一年，PCI DSS 第 10.7 节（PCI，Security Standards Council，2010）；North American Electric Reliability Corporation（NERC）的规则指出了特定日志类型的留存时间；其他法规要求保留特定类型的日志，但是并未指明留存周期。这条原则将帮助你为留存策略打下基础，确定最低需求。

2）评估组织的风险态势：

内部和外部的风险驱动网络不同部分的留存周期。对组织来说，每个风险领域的时间长度以及日志的重要性可能有很大的不同。请记住，如果你的日志主要用于内部威胁调查，则需要较长的留存周期，因为事故常常是多年都未曾发现的，但一旦发现，就必须紧急地将其“追查到底！”

3）关注各种日志来源和生成日志的大小：

防火墙，服务器，数据库，Web 代理服务器，不仅基于需要，也基于典型日志体积及每个日志记录的大小和类型。各种设备或应用程序生成的日志体积有很大的不同。例如，主防火墙会生成大量的日志内容，因此为了满足此数据的长期留存需求，通常只存储 30 天的日志。然而，应该仔细评估某些组织（例如 PCI DSS）的依从性需要以及主防火墙的关键程度，决定是否采用更长的留存周期。对于某些日志来源（例如定制应用程序和不受支持的操作系统），你还可能不具备必要的分析工具。

4）评估可用的存储选项：

日志存储选项包括硬盘、DVD、WORM、磁带、RDBMS、日志特定存储，以及基于云的存储。这方面的决策主要取决于价格、容量、访问速度，最重要的是以合理的周期得到正确日志记录的能力。磁带因便宜著称，然而众所周知，它没有高效搜索存储日志的能力，可能需要人工干预，才能定位合适的磁带和恢复数据以进行分析。还必须考虑，存储媒介的寿命和存储媒介多长时间重写才能满足留存周期需求。例如，7 年的时间对便宜的可写 CD 以及 DVD 可能太长了。技术格式会过时，用于读取磁带和软盘的驱动器可能难以购买到。

表 4.1 是一个主要关注外部安全策略的在线企业的例子。

表 4.1　在线企业留存策略示例（Chuvakin,2011）

类　型	网　络	存储层	留存同期
入侵预防系统（IPS）	非军事区（DMZ）	在线	90 天
防火墙	非军事区（DMZ）	在线	90 天
服务器	内部	在线	90 天

（续）

类　型	网　络	存储层	留存同期
所有	非军事区（DMZ）	存档	3 年
关键	内部	存档	5 年
其他	内部	存档	1 年

组织中的日志留存的留存时间与空间需求是压倒一切的第一需求。本章的其他部分会根据你所在组织的策略，研究各种可能遇到的日志格式以及维护和存储日志文件的策略。

4.3　日志存储格式

在本节中，我们将介绍一些日志存储格式。我们的网络设备、应用程序以及操作系统产生多种不同的日志格式，在许多情况下，日志会被存储为基于文本、二进制或者压缩的格式。

4.3.1　基于文本的日志文件

基于文本的日志记录是目前最丰富的日志类型，这归功于生成这类日志时较低的系统成本，以及现有的许多计算机语言中包含了可以很轻松地生成基于文本日志的框架。许多流行的系统采用了基于文本的日志文件格式，是因为这种格式有着许多的优点。

优点：

- 应用程序写入基于文本的日志文件，从 CPU 以及 I/O 资源来说代价很低。
- 文本格式是典型人类可读格式，可用常规文本工具（如 grep 和 awk，都是各种 Unix/Linux 操作系统变种的固有工具）处理和查阅。
- 许多常见的基于文本的日志格式已经存在，例如 syslog。使得运营和安全团队易于使用一种通用方法来集中化以及解析日志，创造一个更完善的日志管理系统。

1. 扁平文本文件

顾名思义，扁平文本文件在许多方面上是一个扁平的无模式文件，可能遵循某种常见模式或自由格式。系统通常会创建一个新日志文件，并持续向其追加写入，直到磁盘空间不足或某个系统进程（例如我们将在后面讨论的 logrotate）指示系统开始一个新日志文件并将当前文件存档。这种格式倾向于以时间先后排序，最早发生的事件位于文件开始处，最近发生的事件位于文件末尾。

许多系统采用的较常用格式之一是 syslog 格式。在第 1 章、第 2 章以及第 3 章中，我们知道，许多服务器配置为使用 syslog 的 514 端口，通过用户数据报协议（UDP）或传输控制协议（TCP）发送自己的日志数据。通过 syslog 发送的日志在发送之前保存在磁盘上，具有非常简单的特殊格式。下面是一个典型的 syslog 格式的日志消息：

```
May 1 00:14:54 Mac-mini kernel[0]: HFS: Very Low Disk: freeblks: 0,
   dangerlimit: 334
May 1 00:30:47 Mac-mini kernel[0]: add_fsevent: unable to get path for
   vp 0xffffff80154a3d10 (system.log; ret 22; type 4)
May 1 00:30:47 Mac-mini kernel[0]: add_fsevent: unabled to get a path
   for vp 0xffffff80154a3d10. dropping the event.
May 1 06:45:55 Mac-mini kernel[0]: IOSurface: buffer allocation size is
   zero
May 1 15:15:20 Mac-mini kernel[0]: 0 0 AppleUSBCDC: start - initDevice
   failed
May 1 15:15:20 Mac-mini kernel[0]: USBMSC Identifier (non-unique):
   000A2700229F4191 0x5ac 0x1266 0x1
```

使用扁平文本文件长期存储日志数据的显著优势之一是，可以使用大量工具来阅读和审核这种格式的数据。每个平台都有许多工具可以轻易访问和读取此种格式的文件。如果你在未来 5、7 或 10 年需要去阅读或审核数据，需要能够处理和关联事件记录的工具，这一点就变得非常重要。

syslog

syslog 是一个应用非常广泛的日志记录格式，是许多系统（包括 Unix/Linux）的标准日志记录格式。虽然它已广泛使用，但一直并未正式标准化。由于缺少标准，各个系统的实现可能有很大的不同，这也使得跨平台关联事件在某些环境下变得困难重重。syslog 较为常见的使用方式是 BSD syslog 协议，在 http://www.ietf.org/rfc/rfc3164.txt 定义。尽管这并不是一个真正的标准，但是这一参考文献可以用来关联遵循这个 RFC 的不同系统上的数据。

2. 索引扁平文本文件

组织在扁平文本文件上很快就会遇到的一个局限性是从扁平文本文件中快速查询、排序、检索关键元素，以通过他们管理的平台发现有意义趋势的能力。此外，随着日志文件迅速跨入 GB、TB 甚至 PB 级别，再使用传统的 grep，awk 和基于文本的搜索工具会让人失去耐心，变成一个极其耗时的过程。

索引文本文件是一种从日志文件中组织数据的方式，它使日志的关键元素能被快速查询。许多组织可能在组织成长和开始集中日志信息时，很快发现需要结构来生成报表，以及在超出留存周期时销毁日志数据，从而开始采用索引扁平文本日志文件。索引扁平文件具备扁平文本文件的许多优点，具备快速数据插入能力，并保持人类可读的数据格式。

扁平文件索引的一个例子是我们将在 15 章讨论的日志留存工具 OSSEC 所使用的一种策略。OSSEC 以下面的目录结构组织它接收到的日志：

```
/var/ossec/logs/alerts/2011
/var/ossec/logs/alerts/2011/Jan
/var/ossec/logs/alerts/2011/Feb
/var/ossec/logs/alerts/2011/Mar
...
```

从上述结构中可以看出，日志根据接收日志事件的年和月组织。因此，如果我们想审核从 2011 年 1 月开始的 Web 服务器日志，我们应该进入“/var/ossec/logs/alerts/2011/Jan”目录，使用我们的分析工具分析 1 月份的日志。

也有许多旨在生成索引，以加快日志搜索和分析的工具。Apache Lucene Core 是较为强大的工具之一，其主页是 http://lucene.apache.org/core/。Lucene 是一个基于 Java 的框架，它能辅助索引的生成，实现完全的文本搜索日志，并且集成到辅助日志搜索及分析的工具中。

4.3.2　二进制文件

顾名思义，二进制日志文件是应用程序生成的机器可读日志文件，需要专有的工具或程序去阅读或处理它们。在各种环境中的较常见二进制日志文件的例子包括 Microsoft Internet 信息服务（IIS）日志以及 Windows 事件日志。在使用大型主机或定制应用程序的许多环境下，日志文件也可能被编码为二进制或机器特定格式，如广义二进制编码的十进制交换码（Extended Binary Coded Decimal Interchange Code，EBCDIC），在 Intel 和 PC 硬件平台需要工具去解码和阅读它们。

二进制日志文件的长期存储会给组织带来许多挑战。在存储和留存二进制日志文件的原生格式之前需要考虑的问题如下：

- 未来 5、7 甚至 10 年后阅读这些日志的工具的可用性。从现在开始的 10 年内，保留一台 Windows NT 服务器来阅读遗留的 IIS 6.0Web 服务器日志，并进行取证分析，几乎是不可能的。
- 二进制日志文件在磁盘空间利用上很高效。然而，无法进行很大的压缩。压缩的二进制文件的大小可能是原文件的 90%。相比之下，基于文本的文件压缩后可能仅占原文件的 10% 左右。与文本文件日志记录相比，二进制文件所需的存储空间会比较大。

4.3.3　压缩文件

大部分生成日志的系统一般会在日志增长到指定大小时，或者在每天、每周、每月配置的时间周期以及其他条件下开始一个新的日志文件（Kent 和 Souppaya，2006）。之前的日志文件通常会重命名，并以未压缩格式存档于系统硬盘中，以便使其易于访问和查询。随着日志文件的老化，某个日志文件与日常报告和日志审核任务的相关性越来越弱，但它对于符合依从性留存周期和执行取证仍然很关键。在许多情况下，我们依然期望日志文件能够快速访问，但是在磁盘上占用较小空间，以使为新增日志记录和其他系统处理需要节约宝贵的存储

空间。在系统中压缩日志文件就是解决这种需求并节省宝贵磁盘空间的一种机制。

在 Unix/Linux 系统中，有个叫 logrotate 的方便实用程序，它会帮助你管理你所保留的日志文件序号，并进一步压缩日志。下面是实现 syslog 消息日志轮转的一个 logrotate 配置文件示例。该实用程序的更多信息可以参阅 http://linuxcommand.org/man_pages/logrotate8.html(Troan 和 Brown,2002)

```
# sample logrotate configuration file
compress
/var/log/messages {
      rotate 5
      weekly
      postrotate
         /sbin/killall -HUP syslogd
      endscript
}
```

让我们研究一下这些设置，以及它们是怎样帮助我们轮转、压缩和维护 syslog 事件的。表 4.2 对 logrotate 配置脚本进行了分析。

表 4.2　logrotate 压缩与留存 syslog 日志消息所用的设置示例（Chuvakin,2011）

设　置	配置值	注　解
Rotate	5	该设置在从系统中删除日志文件之前将保留最后 5 个消息文件
Weekly	N/A	日志将每周轮转一次。logrotate 中的其他选项是每月和每天轮转
postrotate/endscript	/sbin/killall-HUP syslogd	允许脚本在日志轮转发生之后运行。在这个例子中，我们需要重启 syslogd，使其开始一个新的日志文件

Unix/Linux 系统的许多标准工具都有等价的压缩工具集。zgrep 和 zcat 工具可以从压缩文件中读取和检索数据，就像它们的“同辈”grep 和 cat 在未压缩文件上所做的一样。然而，也有一些工具本身不能处理压缩文件。为了多次利用整个工具集，必须解压该文件，并根据分析需求分配存储空间。在选择一个压缩格式时，为了避免陈旧过时，最好选择一种已经被使用多年并在多个平台可用的格式。在 Unix/Linux 中，tar 和 zip 格式已经使用很长时间，而 PKZip 格式的压缩文件是 Windows 中常见的压缩格式。就像二进制文件一样，我们应该限制未来用于解压和访问日志数据的工具集，避免过时。

4.4　日志文件的数据库存储

到目前为止讨论的许多存储技术需要直接访问系统和特定的日志审核工具集。这样做是很快速和高效的，但是在许多情况下，我们创建摘要报告、过滤器，以及在多台主机间关联日志数据的能力常常受到严重的限制。许多组织发现将日志信息写入数据库中，对需要日志

信息的利益相关方很有用，这样某种格式的日志信息可以快速搜索和查询，并且便于日志审核过程中前端工具的安装和使用。

4.4.1　优点

使用数据库来实现日志留存主要的优点之一是易用性，你可以使用标准的 SQL 查询快速搜索和检索日志记录。数据库系统具有健全的用户访问和权限系统，而且可能已经是组织备份和恢复计划的一部分。现在有许多标准工具可以查询数据库。这些工具允许使用一个通用的工具集来查询日志数据，而不是使用需要专门知识和权限的平台特定工具。许多编程语言内建了数据库数据处理的支持，可以开发用于日志数据实时查看与分析的前端工具。在第 15 章，我们会讨论一些对直接写入数据库有内建支持的工具。这减少了开发和维护内部专用系统将日志加载到数据库中的需求。此外，这有助于将日志数据集中到一个数据库系统中，而且在第 15 章中讨论的许多工具已经具有前端分析和审核的功能。

4.4.2　缺点

将日志数据存入数据库系统并不能避免本身的一些问题和风险。向数据库中写日志消息会有显著的开销。向数据库中写数据明显比写入本地磁盘文本文件慢，主要是因为网络延迟、SQL 解析、索引更新以及向磁盘提交信息。使用数据库存储日志对磁盘的空间需求也较高，主要是因为实现快速搜索和检索需要大量索引文件，压缩数据的选项也较为有限。数据库系统在一个组织中往往有多种用途，因此需面临数据库故障、维护以及为支持日志记录或其他内部系统而升级时的数据丢失风险。当根据留存策略不再需要日志条目时，数据的销毁也会成为问题。若没有合适的规划或分区，删除日志数据可能花费相当长的时间。日志数据一般非常巨大，因此数据的销毁可能指示数据库系统删除数百万甚至数十亿行，并更新所有被删除数据的索引。

4.4.3　定义数据库存储目标

避免上面提出的各种缺点的关键在于，在将数据移到数据库中时要开发存储目标。应该制定一个计划，使存储需求与我们前面讨论过的组织日志留存计划保持一致。一个好的前期规划，将减少未来维护性停机、数据恢复和重建以及部署和运营资源时的需求。我们将审视一些关键领域，将重点放在开发数据库日志存储计划上。

1. 存储什么？

在设计存储日志到数据库时最关键的问题就是“存储什么？”。当使用数据库作为集中存储库以及日常日志审核分析的在线工具时，你应该保留所有的日志条目以及数据库中的日志字段。在第 15 章中我们会介绍许多 syslog 的替代品，例如 rsyslog 和 syslog-ng，它们具备预先建立的数据库模式和配置选项，可以直接将 syslog 移入数据库存储并且包含 syslog 消息

的所有字段。对于不通用的二进制或应用程序日志来说，要想在数据库中维护日志数据，你可能需要编写自己的实用工具、开发自己的数据库模式。良好的分析以及尽力留存所有的字段，能避免在审核系统关键漏洞时发现缺少全面分析所需的关键信息。

在许多组织中，使用数据库作为主要日志存储的缺点就是从日志安全或基础设施角度来说数据量太庞大臃肿了。也有可能，数据库基础设施无法在不影响客户或其他关键系统的情况下，支持这样的数据容量。在这种情况下，混合的存储方法可能工作得更好，原始的日志数据留存在日志系统来源或集中化 syslog 服务器中，但是以下所列的一项或者多项信息将存入数据库中，以便分析和系统报告：

- 头信息，通常包括某事件发生的时间戳以及事件涉及的 IP 地址，单独存储这些信息在构建趋势信息、确定主机虚报、漏报以及系统间事件趋势的联系时非常有用。
- 消息体，通常就是事件的消息，在数据库中存储这些消息主要是用来构建实时警报系统。例如，可以快速地查询和报告频繁出现相同登录失败消息的情况。
- 分析和总结结果，自定义脚本和工具可能被各个系统使用，以确定整体趋势并对结果进行总结。将这些分析存入集中存储库可以简化整个企业的事件分析报告，并在较低的总体数据库存储和可伸缩性需求的情况下，简化组织的集中审计和摘要报告。

2. 快速检索

在定义了存储内容之后，需要进行一些审核和分析，对数据库进行优化，实现相关数据的快速检索。需要定义的关键项之一就是在日常审核或者对报告和警报的常用查询中被使用的数据库列。存储 syslog 数据时的推荐方法是在如下常见 syslog 字段上创建索引：

- 优先级——消息的严重性或相对重要性。
- 日期和时间——表明事件什么时候发生。
- 生成主机——生成这个事件的系统。
- 消息——事件发生的细节信息。

关于日志数据，反复提到的一个主题就是存储大小会持续增长。即使在数据库中构建了索引和查询优化，但在数万亿行数据中搜索依旧会变得缓慢和繁琐。许多数据库系统支持分区。分区允许逻辑上的单个数据库表分裂成较小的多个区块。在日志数据中，基于日期和时间对数据库表进行分区是一个符合逻辑的做法，这样做可以得到如下好处：

- 改善数据插入速度，因为在较小的物理文件中追加，每次插入时更新的索引文件也较小。
- 查询性能改善，数据库系统在小区块数据中检索和过滤性能明显改善。
- 当某个集合的日志数据超过了留存周期，批处理删除可以通过移除某个数据分区实现。和在单独数据库中单独的删除数据操作相比，这样做显著地改善了日志数据的销毁性能。
- 某些系统允许很少使用的分区迁移到较慢但更便宜的存储选项中。

- 数据库维护时可以允许独立的分区离线。这使得数据库作性能优化时可以不影响插入新的日志数据。

3. 报告

组织通常需要使用日志数据生成报告，以支持外部审计人员和内部利益相关方进行日志数据的审计和审核。通常需要额外的报告表，使报告能够快速和按需生成。这些表中应包含组织关键条目的汇总，这些数字可以通过后端进程定时计算，加快前端工具检索数据的速度。即时生成这些信息可能会耗费大量时间，对报告结构的修改也会带来昂贵的代价。以下是生成总结报告需要考虑的领域，应该尽可能添加到存储系统中，以方便报告：

- 分析结果：通常是多个日志条目的组合，这些条目表示某些需要审核或者采取行动的事件。
- 每个主机按照严重性统计的事件数量：这对于在组织中发现攻击模式或者快速查明故障区域非常有用。
- 基于时间的汇总：许多组织可能需要每日、每周或每月的报告，以便和整个组织分享，基于这些时间周期进行汇总能够自动化和加速这种报告的生成。
- 基于网络或设备类型的报告：网络某些部分可能有不同的报告需求或需要符合不同的依从性框架。PCI 是一个常见用例，只在支付卡处理网段时需要特别的报告和审计需求。这需要额外的报告表，以支持独立的审计需求。

4.5　Hadoop 日志存储

传统数据库作为日志数据系统面临各种挑战，包括对日志数据增长的可伸缩性和高峰活动时需要的存储和系统容量。与传统数据库系统相比，Hadoop 是一个较新的替代方案。传统的数据库系统，利用具备快速 SAN 存储的高端硬件，支持大量的并发用户和请求，并满足数据存储需求。相比之下，Hadoop 系统通常由运行着 Linux 的商用 PC Intel 硬件构成，集群的每个节点只有几 TB 本地存储空间，且没有使用 RAID。一个 Hadoop 集群由一系列从节点和最少一个主节点组成。这样的系统在需要增加空间和容量时，可以简单地向集群中添加新的节点。Yahoo 以及 Facebook 等公司现在都使用 Hadoop 支持它们的应用程序，数以 PB 计的最终用户更新和搜索查询信息可以得到快速地处理和搜索。

4.5.1　优点

Hadoop 有传统数据库系统的许多优点。Hadoop 不用在各个系统上使用平台特定的查询工具，就能对日志数据进行快速检索和搜索。Hadoop 通过将搜索请求分布到集群各个节点，快速寻找、处理和检索结果，从而在数据尺寸增长时很好地实现了可伸缩性。Hadoop 主要用 Java 构建，可以开发工具，实现日志数据的实时查看和分析。Hadoop 在跨越 Hadoop 集群各

节点的 Hadoop 分布式文件系统 (Hadoop's Distributed File System,HDFS) 中，将数据存储为一组有结构的扁平文件。这样的结构使得 Hadoop 可以比传统数据库系统更快地进行文件数据插入。Hadoop 同样具备容错性，通过在集群节点间制作多个备份数据，当一个节点出现故障时，数据仍可以从其他节点检索。

4.5.2 缺点

Hadoop 是一个强大的日志存储系统，但是同样存在许多可能影响组织的缺点。现有的许多日志记录工具对 Hadoop 的直接支持有限。Rsyslog 最近已经添加了向 Hadoop 集群写入 syslog 消息的功能，但是对大部分其他日志源来说，依旧需要开发一个向 Hadoop 写入日志数据的工具。可以直接在 Hadoop 中查询和报告数据的工具集依旧非常有限。组织需要开发和维护定制的前端系统，以便进行实时分析和审核，因为符合这一需求的选择还很少。

4.6 云和 Hadoop

Amazon 最近增加了一个新服务，叫做 Amazon Elastic MapReduce（Amazon EMR）。这个系统结合了 Amazon 弹性计算云（Amazon Elastic Compute Cloud，Amazon EC2）和 Amazon 简单存储服务（Amazon Simple Storage Service，Amazon S3）中的许多服务，以提供一个云端的 Hadoop 实例。这一服务允许组织只对所需的计算能力和存储选项付费。对不想自建和投资内部 Hadoop 实例的组织来说，这个选项是一个不错的替代方案。然而，应该仔细评估是否选择这种选项。最明显的一点是，向云解决方案写入数据的时间将会显著增加，因为它发生的地点与生成日志的主机距离很远。此外，数据留存周期较长的组织可能会发现，由于增长的存储尺寸以及较长的存储时间，成本会变得更高。

4.6.1 Amazon Elastic MapReduce 入门

在下面的小节中，我们会通过一个简单的例子演示向 Amazon EMR 中写入日志数据、使用 Pig 语言的 MapReduce 作业解析日志、评估并审核这个任务的分析输出。要完成这个例子，你可能需要一个 Amazon Web 服务（Amazon Web Services，AWS）账户。第一步是在 http://aws.amazon.com/ 登录一个账户。

4.6.2 浏览 Amazon

要使用 Amazon EMR 来做日志留存和分析，我们将使用多种 Amazon 服务，从云中加载、解析和分析数据。

我们首先从独立服务器中将日志数据加载到 Amazon S3 中，Amazon S3 是一个在线存储 web 服务，它允许我们存储无限量的数据（当然是需要付费的），具备很高的可伸缩性，并

提供数据安全和冗余支持。我们上传到 Amazon S3 的数据会被存储到命名的 bucket 中。bucket 是 Amazon S3 中的一个方法，允许我们为所上传数据创造单独命名的存储区域。

将数据上传至 Amazon S3 之后，我们会使用 Hadoop 的 Pig 语言创建一个 MapReduce 作业（job）来处理这些数据。MapReduce 是一个 Hadoop 框架，用于将处理大数据集的计算工作分布到一组作业中。Amazon EMR 是 MapReduce 的 Amazon 云实现，运行于 Amazon Web Services（AWS）框架上。我们将使用 Pig 及其 Pig Latin 任务语言编写 MapReduce 作业，操纵我们的日志数据。Pig 是来自 Apache 项目的实用工具，是 Hadoop 项目的子项目之一，它利用简单的 Pig Latin 语言帮助创建 MapReduce 程序。Amazon EMR 已经包含 Pig，以帮助我们处理将要上传至 S3 数据基础设施的日志数据。图 4.1 中的概略框图表明了我们例子中涉及的处理过程流。

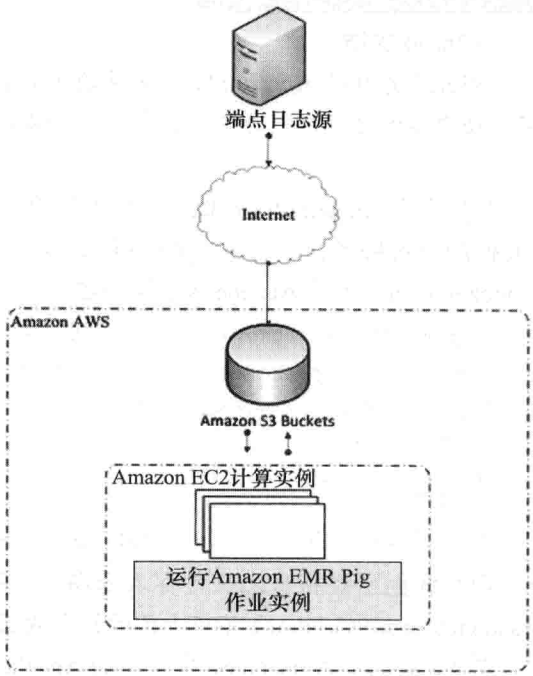


图 4.1　Amazon Elastic MapReduce 日志处理示例略图

Amazon Web 服务账户

收费和你的账户

本节阐述的例子和步骤会在你的 Amazon Web 服务账户中发生费用。在 Amazon S3 bucket 中存储日志文件需要付费，在 Amazon EC2 处理日志所需的执行时间都需要计算成本。运行这个例子的开销取决于你使用的时间和存储容量。在开始运行和体验下面例子之前，可以从这里了解有关 Amazon Web 服务的价格详情：http://aws.amazon.com/pricing。

4.6.3　上传日志到 Amazon 简单存储服务（S3）

我们需要将日志数据加载到 Amazon S3 中，这样就可以使用 Pig 脚本处理数据。上传数据到 Amazon S3 有许多方法，Amazon 还为那些期望在自己现有应用程序中集成 Amazon S3 存储的组织提供了一个 API。在我们的例子中，将使用第三方命令行工具 s3cmd。

S3CMD 和 AMAZON

s3cmd 支持

s3cmd 并非是 Amazon 官方支持的工具，并且当前还处在 beta 版时期。在该工具安装、使用和配置过程中的更多细节问题可以访问该工具的官网 http://s3tools.org/s3cmd。

我们会在 Amazon S3 存储控制台中创建一个叫 "apache-accesslog-buket" 的 bucket。这就是我们的 web 服务访问日志、pig 解析脚本以及结果报告的存放地点。使用如下命令可以创建 bucket 并向它上传 Apache Web 服务器访问日志。

```
$ s3cmd mb s3://apache-accesslog-bucket
Bucket 's3://apache-accesslog-bucket/' created
$ s3cmd put /var/log/httpd/access_log s3://apache-accesslog-bucket/
   accesslog
/var/log/httpd/access_log -> s3://apache-accesslog-bucket/accesslog [1
   of 1]
1203 of 1203 100% in 0s 1825.23 B/s done.
```

现在我们已经将日志文件存入云端了。我们可以在 S3 管理控制台中（https://console.aws.amazon.com/s3/home）审核上传的日志数据。

图 4.2 展示了 Amazon S3 中上传的 Apache 访问日志。

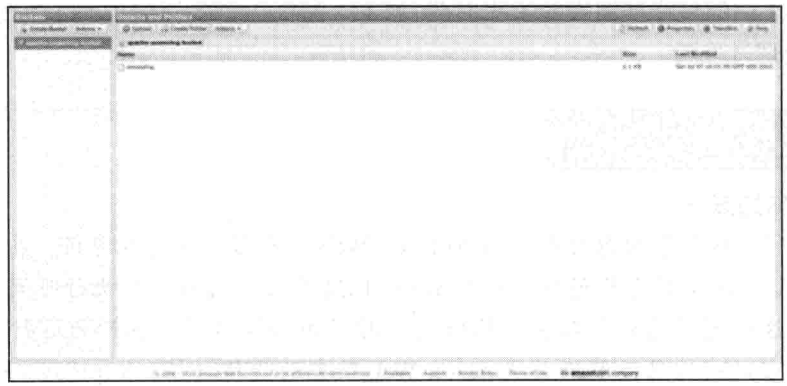


图 4.2　加载到 Amazon S3 的 Apache 访问日志

这是向 Amazon S3 存储上传数据的一个简单例子。s3cmd 命令行工具可以集成到我们在本章开始时讨论的 logrotate 脚本中。这种方法有助于在轮转日志文件时，自动化从终端服务器加载日志信息到 Amazon 云的过程。Amazon S3 对 syslog 馈送没有原生的支持，但是 Amazon 有一个扩展的 API，可以用它来构建你自己的实用工具，另外，在 15 章中我们会讨论一个 syslog 的替代方案——rsyslog，它内建 Hadoop 支持，可以用于私有 Hadoop 基础设施，也可以扩展以便支持 Amazon EMR。

4.6.4　创建一个 Pig 脚本分析 Apache 访问日志

我们期望解析数据，得到表明环境中究竟发生了什么的情报。在 web 访问日志中，研究那些生成大量 HTTP 访问错误的远程主机往往很有用。访问错误和其他 HTTP 错误都有一个状态码，通常是 400 或更大的数字。我们可以用 Pig 的 Pig Lation 语言创建脚本，来解析我们上传的日志，并提供产生此类错误的远程主机的 IP 地址计数。下面的 Pig 脚本会执行这项繁重的任务，生成相应的报告。

```
--
-- setup piggyback functions. Useful pig function library
--
register file:/home/hadoop/lib/pig/piggybank.jar
DEFINE ApacheCommonLogLoader
org.apache.pig.piggybank.storage.apachelog.CommonLogLoader();
--
-- import log file and parse into individual components
--
logs = LOAD '$INPUT' USING ApacheCommonLogLoader as (remoteHost,
   hyphen, user, time, method, uri, protocol, statusCode,
   responseSize);
--
-- Filter Apache GET requests from the log where the HTTP status code
   is 400 or greater
--
logsfiltered = FILTER logs BY method == 'GET' AND statusCode >= 400;
--
-- Retrieve the remote host list that generated the HTTP errors
--
logsremoteaddr = FOREACH logsfiltered GENERATE remoteHost;
--
-- Group parsed information by the remote hosts performing GET requests
--    and create counts on the number of times the host generated an
   error
--
groupedByRemote = GROUP logsremoteaddr BY remoteHost;
statusCounts = FOREACH groupedByRemote GENERATE group AS remoteHost,
   COUNT(logsremoteaddr) AS numStatus;
--
--Store HTTP status error report to S3
```

```
--
STORE statusCounts INTO '$OUTPUT';
```

要使用这个脚本，必须将 Pig 脚本上传到 Amazon S3。为了在后面的小节中使用，我们上传该脚本到 Amazon S3 bucket：s3://apache-accesslog-bucket/HttpStatusCodeReport.pig。

4.6.5 在 Amazon Elastic MapReduce(EMR) 中处理日志数据

既然我们已经将数据和报告脚本上传至 Amazon S3 中，就可以创建一个 Amazon EMR 作业来处理我们的日志数据了。我们可以在 Amazon EMR 的控制台 https://console.aws.amazon.com/elasticmapreduce/home 中创建新作业，在 Elastic MapReduce 控制台中选择“Create New Job Flow”（创建一个新作业流），定义一个新作业，以处理我们的 Apache 访问日志。我们会创建一个叫做“Apache Log Parse Job”的作业，并使用我们自己的 pig 应用程序来生成一个报告。

了解更多关于 PIG LATING 的知识

Pig Latin

要学习更多有关 Pig Latin 语言的知识，可以去这里看看 http://pig.apache.org/docs/r0.7.0/piglatin_ref1.html。

图 4.3 说明如何创建一个新的 Amazon EMR 作业流。

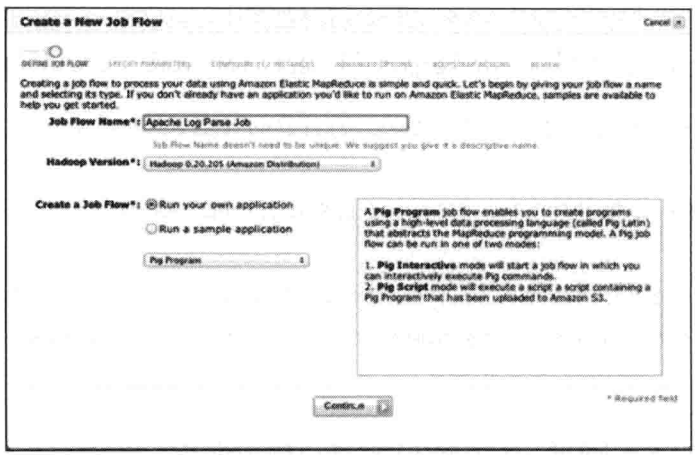


图 4.3　创建一个新的 Amazon EMR 作业流

下一步我们会使用 pig 脚本 HttpStatusCodeReport.pig 来处理日志。我们需要设置一下参数，包括运行的脚本、解析的日志文件以及预期的报告存储位置。

图 4.4 说明如何配置 Amazon EMR 作业流参数。

图 4.4　Amazon EMR 作业流参数

在我们的 Amazon 弹性计算（EC2）实例配置中，我们将配置最低的设置，只使用一个实例。对处理数 GB 日志的较大环境，你可能打算配置多个作业以处理不同日志类型，用多个实例应付每天的日志容量。在你自己的环境中，这可能需要一些实验和研究，幸运的是，Amazon EMR 具备良好的可伸缩性和灵活性，能够适应更广泛的计算需求。对于剩下的设置参数，我们在这个例子中保留默认设置。最后，我们会选择“ Create Job Flow”来创建并开始处理我们的访问日志。

图 4.5 展示了运行中的 Amazon EMR 作业流实例。

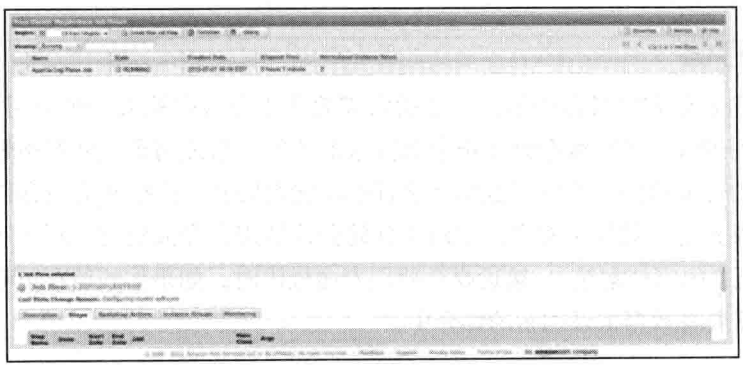


图 4.5　运行 Amazon EMR 作业流实例

当我们的作业流运行完成后，我们在 Amazon S3 中的“ access-error-by-ip”文件夹中生成了所需的报告。本例所上传的日志文件生成了如下报告：

```
192.168.0.6 3
192.168.0.25 3
192.168.0.26 1
```

从结果中，我们可以看到 IP 地址为 192.168.0.6，192.168.0.25 以及 192.168.0.26 的远程主机生成了 HTTP 状态错误，以及各台主机发生的次数。

4.7 日志数据检索和存档

日志的物理存储有多种可用选择。审核这些选择的关键之一是检索和访问各种媒介的速度。通常来说，较高的访问速度选项所需的成本也会更为昂贵。在日志管理中，我们讨论在线、近线和离线日志存储，以及他们相对的成本和访问速度。

4.7.1 在线存储

在线日志信息可以立即访问和检索。这类存储通常是最昂贵的选择，因为必须开启专用硬件的电源供即时检索。在线存储可以是物理连接到服务器的硬盘、数据库系统或存储区域网络（SAN）系统。在线存储系统的典型成本为每 GB 大概 1.3 美元。

4.7.2 近线存储

近线存储是介于在线存储和离线存储之间的存储选择。近线系统通常不需要人工介入，数据作为自动光存储设备或者自动磁带系统的一部分进行检索。这类存储设备的访问时间通常比较长，根据具体系统，从几秒到几分钟不等。近线存储系统的成本范围也很大，但通常只有在线存储系统的一半。借助于添加磁带或光盘，近线存储也有很高的可伸缩性。

4.7.3 离线存储

离线存储是最便宜也最慢的选择，它通常需要人工介入来检索一个可用的光盘或磁带，并将数据恢复到近线或者在线存储系统中以便数据访问。通过购买额外的磁带和光盘，离线存储具有很高的可伸缩性，其价格也通常比近线存储更便宜。近线和离线存储都面临的一个问题就是存储媒介的有效使用寿命。CD/DVD 的公认使用寿命大概为 2 到 5 年，磁带的生命周期也与这相近（国家档案馆）。接近媒介使用寿命极限时，如果你的留存策略要求比媒介更长的周期，你需要将数据重录到新的媒介中。

4.8 小结

从日志记录的方面来说，日志存储是重要的关注点。本章向你介绍了围绕日志存储的各

种相关概念，并给出了许多例子，包括对 Amazon 的 Hadoop 接口的介绍。

参考文献

Chuvakin, A. (2011). SANS Sec434 Log Management Course Material, slide 108.

Kent, K., & Souppaya, M. (2006). Guide to computer security log management. <http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf>.

National Archives. Frequently asked questions (FAQs) about optical storage media: Storing temporary records on CDs and DVDs. <http://www.archives.gov/records-mgmt/initiatives/temp-opmedia-faq.html>.

PCI Security Standards Council (2010). *PCI DSS v2.0*. Retrieved July 7, 2012 from PCI Security Standards, October. <https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf>.

Troan, E., & Brown, P (2002). logrotate. <http://linuxcommand.org/man_pages/logrotate8.html>.

Chapter 5

第 5 章

syslog-ng 案例研究

本章要点

- 获取 syslog-ng
- 什么是 syslog-ng？
- 部署示例
- syslog-ng 故障排除

5.1 概述

你可能会问："为什么我需要一个新的或者不同的 syslog 守护进程？"这是一个好问题。多年以来，在 Unix 系统上的标准 syslog 协议已经足够好了。随着时间的推移，用户、应用程序和组织的需求出现了变化，明显需要某些新的功能。开始，syslog-ng 的出现是为用户提供比现有 syslog 守护进程更多的选项和特性的一次尝试。经过多年的成长，它已经从开源软件变成完全成熟的商业级软件，具备各种不同级别的企业级支持选项。本章将提供 syslog-ng 较为实用的一些功能的概要情况，并讨论一个实际的部署案例。应该注意的是，syslog-ng 有免费版本和企业版本。企业版本不是免费的，它具有免费版本中无法使用的一些功能。

5.2 获取 syslog-ng

获取特定平台 syslog-ng 最新版本的最佳途径是访问其官网：http://www.balabit.com/

network-security/syslog-ng/opensource-logging-system/downloads/upgrade。

提供逐步安装指南确实超出了本章的范围。较早版本的 syslog-ng 需要安装补充的库和工具。现在的 syslog-ng 在单一下载中捆绑了所有需要的内容。syslog-ng 网站包含许多信息来源，帮助你安装、设置和管理：http://www.balabit.com/network-security/syslog-ng/opensource-logging-system/support/documentation/。

5.3 什么是 syslog-ng

正如我们之前所强调的，经典的 syslog 有多方面的问题。例如，使用不可靠的 UDP 作为传输协议。但是有时候，为了 UDP 的无阻塞和低开销特性，放弃传输保证也是一个明智的决定。上述特性可能比 TCP 的特性更重要，因为 TCP 是面向连接的，可能带来一些开销。

另一个问题是安全性。标准的 syslog 协议不支持日志数据的安全传输，对身份认证也没有规定。

syslog-ng 就是试图解决上述问题而不“改变世界”的选择之一。syslog-ng 上值得注意的特性包括：

- TCP 支持：这允许 syslog 数据更可靠地传输。
- 传输层安全（TLS）：syslog-ng 原生支持 TLS 协议。这一特性能够安全传输 syslog 消息，并且可以实现 syslog 消息发送者和接收者来源的相互认证。在今天的环境中，日志数据不能被篡改，而且由于监管依从性问题、法律问题等，往往必须保持机密性，因此上述特性是至关重要的。应该注意的是，TLS 只能用于 TCP 上。
- 数据库支持：syslog-ng 支持向远程数据库发送 syslog 消息。版本 3.3 支持 MySQL、Oracle、MSSQL、PostgreSQL 和 SQLite 3。这一功能的主要优点之一是允许日志数据保存在关系数据库中，使得报告、搜索和其他分析技术非常容易实现。但是，缺点是数据库中没有保留原始日志消息，如果刑事检控中需要你的日志数据，这对于证据链有影响。
- IPv4 和 IPv6 支持：syslog-ng 支持从两类网络上发送和接收 syslog。
- 灵活的过滤：syslog-ng 能够过滤 syslog 消息中的不同字段。从过滤中创建宏，可以用于将消息写入文件或者目录。
- 消息重写：可以创建规则，重写 syslog 消息中的字段。例如，你可以补充消息中丢失或者不完整的数据。
- 消息主体处理：syslog-ng 允许你处理和解析实际的日志消息。syslog-ng 的当前版本（3.3）能够分离 CSV 格式化后的日志消息的各个列，也就是说，由逗号、制表符或者其他分隔符分隔的列，例如 Apache 日志。
- 高性能：syslog-ng 针对高速消息处理，当然，这取决于具体的配置和处理需求，它能够可靠地以每秒数十万条的速度处理消息。

我们来举一个 syslog-ng 更高级特性的例子。它能够用快速和灵活的模式匹配引擎处理消息，从消息主体中提取有用信息。例如，OpenSSH 服务器的登录消息有如下的形式：

```
Accepted password for joe from 10.50.0.247 port 42156 ssh2.
```

这可以用如下的 syslog-ng 模式描述：

```
Accepted @QSTRING:authentication_method: @ for @QSTRING:username:
   @ from @QSTRING:client_address: @ port @NUMBER:port:@ @
   QSTRING:protocol_version: @
```

可以看到，日志消息变化的部分用特殊变量解析。已解析消息各个部分所指定的名称变成 syslog-ng 宏，可以用于过滤器、文件名、数据库列名等。这种模式匹配方法与其他方法（例如正则表达式）相比的好处是速度更快、可伸缩性更好，模式更容易阅读、理解和维护。如果日志消息匹配某个模式，也可以触发动作：经过某些调整，这可以用于在检测到特定事件时发出警报。这一特性的完整参考信息参见 syslog-ng 管理员指南中的《Processing message content with a pattern database》。

这些特性和许多其他功能在《3.0 Administration guide》中有详细介绍：http://www.balabit.com/sites/default/files/syslog-ng-v3.0-guideadmin-en_0.pdf。

5.4 部署示例

这里介绍的示例要部署三个组件：

1）通过 UDP 向本地收集服务器发送 syslog 的客户端机器。客户端机器运行现有的 syslog 守护进程。

2）从客户端机器收集 syslog 的本地收集服务器。这个系统运行 syslog-ng。

3）通过启用 TLS 的 TCP，从本地收集服务器上接收日志的全局收集服务器。

图 5.1 描述了这一部署。

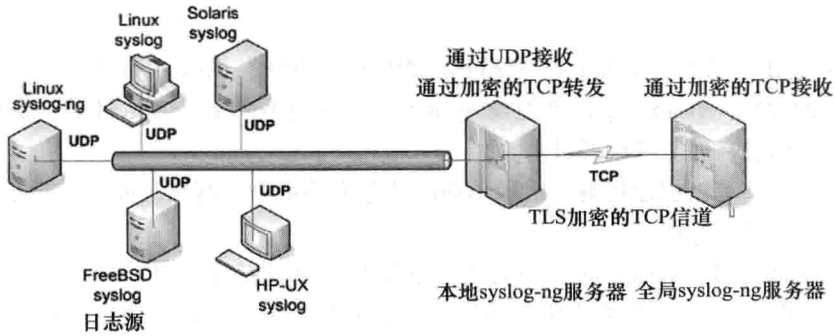


图 5.1　展示我们的部署示例

本地 syslog-ng 服务器将把所有收集的数据传送给远程站点，该站点上安装了另一个

syslog-ng 服务器。这样的站点可以作为全局集中站点、备份的“日志落地点”或者其他环境，通过 WAN 连接到我们的 syslog-ng 收集服务器。因此，日志数据传输需要加密保护。实现这一点的最简单途径是使用 syslog-ng 内建的 TLS 支持。

所以，在两台 syslog-ng 服务器上部署 syslog-ng 之后，就可以开始配置所有系统的日志记录了。

配置

本节我们介绍配置上述设备所需要的基本知识。配置 syslog-ng 的完整论述超出了本章的范围，可以参考 syslog-ng 管理指南，获得完整的使用细节。

为了保持一致，我们将把通过 UDP 收集日志的集中日志服务器称为 10.10.10.1，远程日志服务器称为 10.11.11.1。在这一场景中，我们假定将收集所有日志。在实际工作中，可以在多个位置上应用过滤器，控制数据的泛滥。

1. 日志源

对于运行常规 syslog 的现代 Linux 系统，我们只需要在 /etc/syslog.conf*.* 中有如下一行：

```
etc/syslog.conf*.*              @10.10.10.1
```

syslog 守护进程需要重启，通常由如下命令完成：

```
/etc/init.d/syslog restart
```

对于较老的 Solaris 7 或者 8 系统，更改方法也类似：

```
*.debug      @10.10.10.1
```

然后通过如下命令重启：

```
/etc/init.d/syslog stop ; /etc/init.d/syslog start
```

对于最新的 FreeBSD 系统，更改也类似。

对于最新的 OpenBSD 系统，更改类似于 Linux：

```
*.*      @10.10.10.1
```

重启命令如下：

```
kill-SIGHUP 'cat /var/run/syslog.pid'
```

对于 HP-UX 系统，更改如下：

```
*.debug     @10.10.10.1
```

然后重启。

最后，syslog-ng 需要实现如下的配置文件：

```
source src { unix-stream("/dev/log"); internal(); };
destination loghost { udp("10.10.10.1"); };
log { source(src); destination(loghost); };
```

在已有 syslog-ng 配置的情况下，必须合并上述配置。你可能已经猜到了，守护进程（在这种情况下是 syslog-ng 守护进程）需要重启。

2. 本地 syslog-ng 服务器

这个系统将从各种不同的来源接收 UDP 消息，并将通过基于 TCP 的隧道转发给其他机器。TLS 是实现这种隧道的最佳方法（Schroder，2010）。下面列出启动这一功能所需的基本步骤：

1）用 openssl 之类的工具创建一个 X.509 证书。

2）将证书复制到本地 syslog-ng 服务器。

3）使用 openssl 之类的工具，根据 X.509 证书创建一个哈希文件。

4）创建一个使用如下 syslog-ng 配置块的 TLS 配置语句。

```
destination demo_tls_destination {
tcp("10.11.11.1" port(6514)
tls(ca_dir("/opt/syslog-ng/etc/syslog-ng/ca.d"))); };
```

《3.0 Administration guide》的 3.13 小节提供了这方面和 syslog-ng 所有特性的更多细节。

3. 全局 syslog-ng 服务器

全局服务器将接收用 TLS 加密的消息。配置其接收这种消息的基本步骤如下：

1）将 X.509 证书复制到服务器。

2）复制证书私钥到服务器。

3）配置全局服务器接收加密的日志消息。

```
source demo_tls_source {
tcp(ip(0.0.0.0) port(6514)
tls(key_file("/opt/syslog-ng/etc/syslog-ng/key.d/syslog-ng.key")
cert_file("/opt/syslog-ng/etc/syslog-ng/cert.d/syslog-ng.cert"))
); };
```

同样，更多细节参见《3.0 Administration guide》的 3.13 小节。

4. 数据库日志记录

全局 syslog-ng 将接收到的日志消息拷贝转发给一个 MySQL 数据库。该数据库作为从远程日志收集点接收消息的结构化和可搜索存储。这种设置的主要好处是分析和搜索可以在不影响远程系统的情况下执行，也不需要远程登录到这些系统，手工搜索日志文件。第二个好

处是你还可以有另一个日志数据存储地点。这在全局收集服务器崩溃或者不可用的情况下很理想。

下面是用于 MySQL 的一个 syslog-ng 配置示例：

```
options {
   use_fqdn(yes);
};
destination d_sql {
   sql(
         type(mysql)
         host("127.0.0.1") username("root") password("")
         database("logs")
         table("messages_${FULLHOST_FROM}_${R_YEAR}${R_MONTH}${R_DAY}")
         columns("datetime", "host", "program", "pid", "message")
         values("$R_DATE", "$FULLHOST_FROM", "$PROGRAM", "$PID",
   "$MSGONLY")
         indexes()
   );
};
log {
   destination(d_sql);
};
```

下面是值得指出的几点：

- 数据库名称是 logs。
- syslog-ng 将自动创建一个名为 messages_${FULLHOST_FROM}_${R_YEAR}${R_MONTH}${R_DAY} 的表，这意味着每台发送日志消息的主机有自己的一组表，按照发送日志消息的年、月、日分段。
- syslog-ng 还将为该表自动创建列，也就是 datetime(日期时间)、host(主机)、program(程序)、pid（进程 id）和 message（消息），默认情况下，每个列使用的 SQL 类型是文本。你可以覆盖默认类型。
- 在配置的 options 部分指定了 use_fqdn(yes)；如果没有设置这个选项和使用 DNS，你的消息源将是一个 IP 地址，只记录 IP 地址的第一个八进制数。指定了这个选项，在表名和记录本身都将使用完整的 IP 地址。

《3.0 Administration guide》的 8.2.5 小节详细讨论了这一特性。

5.5 syslog-ng 故障排除

最值得关注的是 syslog-ng 丢包的现象（Open Source Information Security，2010），这可

能表示 syslog-ng 服务器入站消息超载。为了监控这种情况，首先必须启用 syslog-ng 的统计数字收集。控制这一行为需要如下全局配置选项：

```
options {
   stats_freq(10);
   stats_level(2);
};
```

第一条语句开启 syslog 消息形式的统计数字生成。传递给该选项的变量是 syslog-ng 把消息写入配置中指定的任何目标的秒数。这里，我们每隔 10 秒发出统计数字。第二选项指定级别。级别共有三种：1、2 和 3。默认级别为 1，但是为了获得我们需要的特定数据，应该指定级别 2。下面是 syslog-ng 生成的统计数字消息的一个例子：

```
Nov 30 12:32:35 172 syslog-ng[1343]: Log statistics;
   processed='source(s_net)=0', processed='src.file(s_local#2,/
   proc/kmsg)=0', stamp='src.file(s_local#2,/proc/kmsg)=0',
   processed='source(s_local)=1', processed='src.host(172)=1',
   stamp='src.host(172)=1291138345', processed='src.udp(s_
   net#0)=0', stamp='src.udp(s_net#0)=0', processed='src.
   internal(s_local#0)=1', stamp='src.internal(s_
   local#0)=1291138345', dropped='dst.file(d_messages#0,/
   var/log/messages)=0', processed='dst.file(d_messages#0,/
   var/log/messages)=1', stored='dst.file(d_messages#0,/
   var/log/messages)=0', processed='center(received)=0',
   processed='destination(d_messages)=1', processed='center(queu
   ed)=0'
```

我们感兴趣的是“dropped=”条目。它表明对于给定的目标，系统不得不抛弃任何消息。你在配置中指定的所有目标在此都会显示。

启用上述功能也有缺点。按照 syslog-ng 手册中的说明，级别 2 和 3 可能导致 syslog-ng 进程对 CPU 和内存的使用。在此基础上，启用这些功能需要重启 syslog-ng，如果你运行于高可用性环境，这一点可能不现实。检测 UDP 数据包是否被丢弃有一种替代方法：使用 netstaty 命令：

```
# netstat -us
Udp:
   169412 packets received
   170 packets to unknown port received.
   0 packet receive errors
   167002 packets sent
#
```

这条命令的输出显示是否有 UDP 数据包被丢弃。这是故障排除中简单而有效的一种方法。

5.6 小结

syslog-ng 提供了许多高级功能，将日志数据传输和收集提升到新的水平。syslog-ng 具备 TCP 传输支持、TLS 和数据库日志记录等功能，是现成的 Unix syslog 守护进程（缺乏功能，提供的价值不大）的优秀替代品。本章介绍的部署示例简单地展示了现实世界中的配置。

参考文献

Schroder, C. (n.d.). Build a secure logging server with syslog-ng (Part 2). *EnterpriseNetworkingPlanet—News, trends and advice for network managers*. N.p. Web. November 29, 2010. <http://www.enterprisenetworkingplanet.com/netsysm/article.php/3598146/Build-a-Secure-Logging-Server-with-syslog-ng-Part-2.htm>.

Open source information security: Syslog-NG performance tuning. *Open Source Information Security*. N.p., n.d. Web. November 29, 2010. <http://15cards.blogspot.com/2007/11/syslog-ng-performance-tuning.html>.

Chapter 6

第 6 章

隐蔽日志

本章要点

- 完全隐藏的日志设置
- 使用 Honeypots 记录日志
- 日志记录的隐蔽通道简介

恶魔通常藏于暗处
好人也会么？
当然会，为了特定的目的
Haiku 8:Anton

6.1 概述

常识和各项法规制度迫使我们将工作的重点放在保护日志免于各种攻击上，其实还可以使用一种保护方法就是将日志文件和日志记录基础设施隐藏起来，使试图破坏的攻击者无法看到。加密、基于安全代理的日志收集以及完整性检查是有效的（而且，在许多情况下，也是强制的），在攻击者面前隐藏自己的日志记录和日志收集架构有一种独特的吸引力，那就是在安全日志记录这场“战役”中，你可以“兵不血刃”地取胜。毕竟，如果攻击者发现他正被秘密地记录着，那么你只能退而求其次，使用加密以及其他措施。但是，多年实施“蜜罐”（honeypots）甚至整个蜜罐网络（honeynet）的经验使作者相信，大部分攻击者都没那么幸运。

这种掩护下的基础设施是安全技术的一个罕见特例——防御者可能获得优势。

本节介绍这种特殊的方法，因为它的重要性、与其他技术结合的能力、在“蜜罐”中的使用以及围绕隐藏日志记录的一些谜团。

隐藏日志记录的想法是很有吸引力的，尤其是有时它可以解释为：若我们可以对潜在的攻击者隐藏日志收集组件，那我们还需要安全、加密保护日志收集组件么？它们会完全不受攻击的影响。另一方面，其他一些人对“隐藏安全性”有争议，他们说隐藏日志记录是徒劳的，因为攻击者总是能找出它们。像往常一样，两种极端的观点距离真理都很遥远。

为了让这一切清晰明了，我们短暂地离开主题，解释一下为什么“隐藏安全性”被认为是不好的安全性。首先，“隐藏安全性”实际上应该称为“只使用隐藏方法的安全性”。如果你的保护措施完全依赖于攻击者不知道基础设施的关键信息，例如使用的安全措施、安全技术等等，那么一旦他们知道这些后会怎样呢？

安全城墙轰然倒塌！另一方面，环境的安全性可以通过隐藏重要信息而得到显著增强，但仍然不能依赖于它们的私密性。我们来看一个简单的例子：某台计算机并未连接到互联网，并且有武装警卫站在每个终端附近。这是否意味着你必须使用即时贴来“广播”你的密码？不，保持密码的私密仍然是有意义的，即使它们的泄露不会破坏你的安全性。

类似地，隐藏日志记录基础设施的各个部分有助于改进你的安全性，补充其他安全措施，如加固的日志服务器、加密隧道和可信访问控制。

现在，让我们回顾一下什么日志可以被隐藏，以及将它们藏于何处。我们会从一个深刻的“隐藏”日志记录例子开始，这个例子是本书作者之一在安全领域职业生涯的早期中遇到的，并在后来加以利用。它对于手边的工作出奇地有效。

众所周知，业余攻击者闯入 Unix 和 Linux 系统后，会首先结束 syslog 守护进程，阻止本地添加新增日志记录和向远程日志服务器传送日志。这可以通过类似“killall syslogd”的命令实现。所以，在这个小公司网络中，系统管理员将编译了 syslog 守护进程的另一个拷贝，命名为“cachefd”（或者其他相似的模糊的“Unix 风格”进程名称）并且用管理员主目录下的配置文件运行。该文件配置为将日志累积到一些非标准的目录下（例如 /usr/src/kernel）。这样，能够有效地阻止业余攻击者（我们认为这是该公司面临的主要威胁，正如我们所知，对于现在的大部分公司也是如此）终止日志记录。当然，这种方案取决于公司确实关注日志文件，并且有周期性的日志审核，或者具备准实时的日志监控能力。

现在，高级的安全领域从业者（包括作者）可能会嘲笑这种“安全日志记录”，但是在“蜜罐”中观察到的业余攻击者的方法都没有超出结束 syslog 守护进程，并且从没有寻找其他日志记录机制。因此，以上的措施实际上 100% 有效。

我们以对 syslog 隐身嗅探器的经典例子，开启隐藏日志记录的旅程。这个范例被一位作者用在蜜罐网络中，据报道，它还曾经有效地用于生产环境（例子参见 Bauer，2002）。

6.2 完全隐藏日志设置

在这一章中，我们将讨论通过 syslog 和使用嗅探器获取日志的完全隐藏日志设置，覆盖了日志生成、转移和采集等内容。

6.2.1 隐藏日志生成

隐藏日志生成提出了一个挑战，因为生成日志的程序非常可能存在于一个被入侵的信息资产上。通过包捕获采集的日志几乎都是隐藏的，但是生成日志这一环节很难隐藏起来。

一般来说，有以下两个选择：

- 完全地隐藏日志记录行为；
- 隐藏日志记录，并伪造一个日志记录守护进程，满足攻击者的心理。

显然，还可以采用通过使用内核权限创建无法杀死的日志守护进程等其他技术。

1. 隐藏日志记录

使用内核能力或 Unix、Linux 上的专用可加载内核模块（LKM）来隐藏守护进程是众所周知的。本质上，当我们谈论日志记录的 rootkit 技术时，通常有两个额外的选择：

- 可以隐藏其他运行进程（例如我们的日志记录进程）的单独隐藏模块或者通用隐藏程序。大多数的 rootkit 都基于这一原理。
- 专用隐藏登录模块。Honeynet Project 的 Sebek 工具基于此方法。这种方法也可以隐藏其他程序。

最后要注意一点，如果攻击者采用类似原理的 rootkit，他仍然可能干扰我们对日志记录的隐藏。

2. 利用误导隐藏日志记录

如果我们运用上述技术，并留下默认的日志记录基础设施，让攻击者去杀死它，大部分攻击者都极有可能无法进一步攻击，并认为没有其他的日志机制。

这只需要创建一个单独的 syslog 实例，并连接到基础设施（参见下一小节），保留默认的实例，启用一些功能，使它不会干扰真正的日志记录。

6.2.2 隐藏日志采集

本节介绍设置 NIDS、防火墙或其他基于 syslog 的日志隐蔽采集机制的方法。我们将使用嗅探器采集日志服务器上的日志。

首先，我们要保护什么？这种方法隐藏集中日志收集服务器的身份，甚至将攻击者的目标转移到另一台我们选定的服务器（可能是一个“日志服务器”蜜罐，配置用来记录和分析针对日志服务器的攻击），从而抵御攻击。和上述的隐藏日志记录结合，这种方法还可以抵御

对本地日志记录的攻击。

主要的思路是，我们不使用监听守护进程来采集基于用户数据报（UDP）的 syslog，代之以隐藏接口（换言之，没有配置 IP）的嗅探器。结合一些其他的技术，我们就可以转移攻击者对日志收集服务器中重要日志的兴趣。按照我们对“隐藏安全性”的描述，这并不意味着日志服务器不应该尽可能地加固，也不意味着不应保护其免受未经授权的访问或恶意攻击。

让我们考虑一下，如何为上述的情况做配置：

6.2.3 IDS 日志源

本节我们将配置基于 syslog 日志的目标。我们需要日志转发到某个地方；不是直接发送到收集服务器中，而是发送到一个“虚假”的服务器，或者我们网络中一个随机 IP 地址（不被主机使用的 IP 地址），如果以“蜜罐”作为目标就更好了。

如果我们要发送系统日志或者其他的日志，就需要修改标准的“ /etc/syslog.conf”文件，例如加上：

```
*.*    @fakeserver
```

“fakeserver”代表着我们的诱饵服务器名称或者 IP 地址。也可以加上：

```
*.*    @10.10.10.10
```

10.10.10.10 是同一局域网段中没有主机占用的 IP 地址。

上述方法很可能只在 Linux 和现代 BSD 变种中有效。在旧的 Solaris 系统中，我们需要指定消息优先级（不能像 Linux 一样使用通配符），如：

```
*.crit    @fakeserver
```

如果将流行于 Unix 或者 Linux 上的 Snort IDS 作为伪日志记录源，以上方法有效，因为警报转发是由 syslog 守护进程而不是 Snort 完成的。只要选择 syslog 机制，这种技巧就可以在商用的 Sourcefire Snort 设备上使用。

如果将运行在 Windows 上的 Snort IDS 作为伪日志记录源（不太常见，但是仍然可在某些环境中看到），我们需要修改“snort.conf”配置文件，它将有下面这一行：

```
output alert_syslog: host=10.10.10.10, LOG_AUTH LOG_ALERT
```

因此，在日志服务器上的配置和平常没什么不同。但是，如果攻击者打算通过入侵该系统或 IDS 传感器来攻击日志服务器，就会被引上错误的道路。实际上，“正确的道路”被我们用来研究他们的行为。

6.2.4 日志收集服务器

这是发生大部分重大更改的地方。我们使用嗅探器程序从网络中采集日志。这就需要做

出一些选择。首先：服务器应该有 IP 地址吗？如果有两个网卡（理想情况），我们可以设置一个为管理接口（配有 IP 地址），另一个则是嗅探接口（没有 IP 地址，嗅探器程序在该接口上运行）。这种配置是可管理性和安全性的最佳组合。接下来我们将描述这个配置过程。

在管理接口上，我们通过防火墙配置访问控制（例如 Linux 的 iptables，每个 Linux 系统里都有的标准防火墙代码）。该接口上将运行一些远程访问服务（ssh 是 Unix 远程管理标准）。

在没有 IP 地址的嗅探接口上（通常叫做隐藏接口），我们安装嗅探器程序。选择很多，最主要的一种选择归结为：利用通用嗅探器（例如 tcpdump 报文嗅探器）从线路上采集日志，然后从记录下的流量中手工提取，或者使用专门的嗅探器。

前者的缺点很多。我们需要一些机制，从捕获的数据流量中提取日志并转换为 syslog 等标准格式，然后再将它们和其他日志合并。而且，这些嗅探器用二进制格式记录数据，这就需要将数据转换为文本才能分析它。然而，许多流行嗅探器的安全漏洞经过了很好的审计，可能对暴露在恶意网络下提供更好的解决方案。有人会对“暴露”这个词存在争议，因为似乎攻击者没有办法知道是否有嗅探器从而攻击之。不幸的是，事实并非如此。对隐藏接口上运行的嗅探器发动攻击的方法已经众所周知了，即便是身为所有嗅探器祖师爷的 tcpdump，也逃脱不了这个命运。

利用专用的日志嗅探器，可以很容易地和其他日志记录的基础设施组件集成。例如，我们可以在线下采集 syslog 日志，并马上发送到另一个系统（如分析服务器）中去。

我们将讨论两个此类日志记录方案：plog（由 Marcus Ranum 在几年前编写的）和 passlog（由 Christian Void 编写）。Plog 可以在 http://www.ranum.com/security/computer_security/code/ 获得，而 passlog 则可以从 http://freecode.com/projects/passlogd 下载。

plog 是一个嗅探 syslog 消息流量并转发到 /dev/log（用于转发消息给 Unix 系统 syslog 守护进程的 syslog 设备）中的工具。

虽然我们尽可能不去转载操作手册或者命令行选项清单，但是这些重要工具是个例外。下面是 plog 操作在蜜罐网络系统中的一个范例：

```
# ./plog -i eth1
```

上述命令行启动 plog（它派生出一个守护进程），并且使其嗅探 eth1 网卡上采用 UDP 端口 514 发往任意系统的 syslog 格式消息。上述配置的不足之处是，消息看上去似乎是源于运行 plog 的系统，而不是源于网络中实际的消息源。

plog 用于采集 LAN 中蜜罐服务器发出的日志。这些消息的目标甚至是没有配备 syslog 守护进程的其他系统。这就是为了迷惑攻击者的！。

passlog 是类似的工具，可以用于相同的目的。主要区别在于它拥有有更多的选项，并且具备在线路上捕捉时将消息同源、目标关联的能力。例如：

```
Mon May 5 14:22:42 2001 baby to mama: <23> su[821]: Authentication
   succeeded for user 'anton'
```

在使用 passlogd 之前，你应该记住一点：0.1e 之前的版本存在很严重的缺陷，即使在隐

藏接口上进行嗅探也可能遭到攻击（正如我们上面所提示的）。不仅已知的漏洞，甚至连有效的漏洞利用方法也已经公开发布。

下面是一个被修改的 passlog 嗅探程序用于作者的蜜罐网络中的方式：

```
$ passlogd [...]
```

上述命令捕获从蜜罐受害服务器发往其他系统（必须安装 syslog 守护进程）的所有消息。两种工具都可以安装在日志收集服务器中，用来嗅探日志并立即将日志发往其他地方，或与其他收集的日志文件集成。

关于将日志发送到不存在的服务器，需要注意一个更为重要的意见。为了发送 UDP 报文，发送服务器需要从网络中愿意接受 IP 连接的服务器那里得到响应。这需要通过 ARP（地址解析协议，将 IP 地址映射到 MAC 硬件地址）实现，因此进行一些 ARP 欺骗是合理的。下面说明如何实现 ARP 欺骗：

```
# arp -s 00-05-00-e5-8a-28 10.10.10.10
```

上述命令必须在运行嗅探器的服务器上运行，但是命令中包含的 MAC 地址可以属于网络中任一台机器，IP 地址甚至可以不存在。

6.2.5 “伪”服务器或“蜜罐”

这种特别的服务器并不是必需的，但是安装它是一个很好的主意。原因是，如果攻击者发现没有一台服务器监听着日志，那就一定有某些“好玩”的事情可做，这个环境的安全性大大强于他开始时的猜想（另一方面，业余攻击者可能会以为 syslog 服务器死机，或者是管理员配置错误）。

这类服务器有很多选择。它可以拥有一个最低安全限度的 syslog 守护进程，没有任何其他的程序。在服务器中你可以配置实际的日志采集，也可以不配置，但是它的真正目的是吸引攻击者进入一个陷阱，从而浪费攻击者的资源。

第二个选项是在该 IP 地址上设置一个成熟的蜜罐。听上去是个好主意，因为高级攻击者有可能在入侵向其发送日志的系统之后泥足深陷。创建蜜罐的方针超出了本书的范围，请查考参考文献部分，寻找一些具有该主题信息的书籍。本章我们将提到一些蜜罐日志记录问题，但并不是详细的设置指南。

6.3 在“蜜罐”中的日志记录

虽然本章最后的参考文献会提供许多关于“蜜罐”的资料，但是我们仍然要简单介绍一下蜜罐的定义和背景。

蜜罐（honeypot）是一个虚假的目标机器，用以观察黑客的攻击。蜜罐网络（honeynet）则是一个围绕这样的虚假机器构建的网络，目标是诱惑和跟踪黑客的攻击过程。通过研究

真实世界的攻击，研究人员希望能预测出新趋势，以此开发出更高级的防御体系。Lance Spitzner 是这方面的尝试——Honeynet Project（http://www.honeynet.org）的奠基人之一，他将蜜罐定义为“一种安全资源，价值体现在被探测、攻击或者入侵时。”Honeynet Project 又将蜜罐分为研究型和生产型；前者目标在于获取攻击者的情报和技术，后者旨在降低 IT 资源的风险并为网络基础设施提供攻击预警，还可能使攻击偏离高价值的目标。

正如我们上面提到的，蜜罐就是一个日志被攻击、破坏和阻止的环境；也是通过隐藏日志记录技术来保护日志最有意义的环境，因为其他蜜罐网络基础设施也对攻击者隐藏。另一方面，这种环境下的控制难度更大，因为与生产环境相比，我们需要在蜜罐环境中记录的细节更多。

我们要在蜜罐中记录什么呢?

1）网络事务记录

2）网络流量记录

3）主机活动记录

4）IDS 警报

我们来简单地了解一下上述日志类型。

蜜罐中的网络事务记录包含了网络第 3 层和第 4 层的数据，如源 IP 地址、目标 IP 地址、协议、端口等等。这些数据通常由蜜罐网络防火墙和网络监控工具（如流收集器）以及网络 IDS 捕获。网络事务还包括互联网的入站通信和连接尝试，蜜罐网络内部机器之间的连接以及由蜜罐发起的出站连接（这是最不好的预兆）。

和生产网络相比，网络流量记录日志可以视为蜜罐的专有特征，因为很少有人能够承担对穿越该网络的所有数据包的记录。它还包括原始二进制格式的网络通信完整记录。

主机活动记录包括攻击者活动（例如击键和攻击者对系统的影响）记录和同时发生的其他主机进程活动（标准应用程序和操作系统日志文件）。

蜜罐网络产生的最后一类日志——IDS 警报，是了解蜜罐中到底发生了什么的主要方法。它增加了用于网络流量分析的数据结构，并允许你根据已经发生的事情采取行动。

显然，上面的所有日志都需要保护。

首先，我们要确定即便攻击者已经破坏了系统的标准非隐藏日志记录功能，发生在蜜罐上的事件也都能被记录下来。正如前面所提到的，在大部分的 Unix 蜜罐网络入侵中，攻击者的第一步都是结束日志记录守护进程。

6.3.1 蜜罐网络的隐蔽 shell 击键记录器

Honeynet Project 的 bash shell 补丁是蜜罐日志记录器的一个简单范例（由本书作者之一进行改进）。我们可以从 Honeynet Project 的工具页面（http://www.honeynet.org/tools/index.html）上下载。

该工具可以记录 bash shell 提示符下的击键（这是蜜罐上捕获的信息中最为重要的，因

为它们说明了攻击者在系统上的所作所为），并从蜜罐中转移到收集服务器。为 Windows 世界的用户介绍一下，bash shell 是大多数 Linux 系统的标准命令行解释器，也是远程用户（和许多本地用户）与系统交互的主要机制。

因为我们讨论的是隐蔽日志记录，所以将重点放在这个工具的隐蔽性上。对于蜜罐的“客人”（攻击者）来说，它能提供的保护就是隐藏自己的存在。在蜜罐中被捕获的业余攻击者通常不会怀疑这个系统 shell 是专门用来对付他们的。他们通常禁用历史日志记录，并认为他们的击键不会在任何地方保存下来。

bash shell 直接通过 UDP 协议将命令转发到远程机器上，不使用任何主机程序。bash 自身包含一个 UDP 发送器（通过线路发送数据包）。UDP 通信对于隐蔽日志传输任务是非常合适的（提醒一下，UDP 还用于标准的 syslog 记录器）。UDP 从某种程度上说比传输控制协议（TCP）更难于检测（例如，UDP 通信不总是在 netstat 中显示），不需要“root”权限（不同于更深奥的隐蔽数据传输方法，例如 Internet 控制报文协议（Internet Control Message Protocol，ICMP）的有效载荷，或网络隐蔽信道相关作品中讨论的无连接 TCP 数据传输），因此，即使攻击者没有“root”权限，它也可以直接用于攻击者所运行的 shell。

这种日志记录通道的另一个保护措施是“目标欺骗”（在前面已做了描述）。如前所述，UDP 目标硬编码为一个不相关的服务器甚至另一个蜜罐。注意，这种欺骗指的不是普通的 IP 欺骗（称为“源地址欺骗”），而是向不存在或没有监听的主机发送 UDP 数据报，并用嗅探器采集。这使你可以屏蔽接收日志服务器的位置。既然 UDP 协议是无连接的，将日志数据包发送到未明确选择接收的机器就变得可能了。

下面是一个 bash 日志的例子：

```
T=17:19:15-12222004. PI=11354 UI=0 ./dumbscan -B 10.11
```

格式中包含了命令执行的时间和日期（2004 年 12 月 22 号 17:19），shell 的 UNIX 进程 ID（11354），UNIX 用户 ID（0 表示用户是“root”）以及执行的命令（“./dumbscan–B 10.11”）（试图扫描从 10.11.0.0 至 10.11.255.255 的 B 类 IP 地址，进行攻击）。

这种隐蔽日志记录器也可用以监控生产环境中的用户。

这种日志记录机制的隐蔽性并不高，因为它很容易被蜜罐的攻击者嗅探到。而且，通过观察 bash shell 二进制文件也可以检测出来。为了解决这类问题，Honeynet Project 提出了名为 Sebek2 的隐蔽日志记录器，我们在下一节简要介绍这种记录器。

6.3.2 蜜罐网络的 Sebek2 案例研究

Honeynet Project 的 Sebek2（http://www.honeynet.org/papers/sebek.pdf）是隐蔽记录器的最佳范例，具有隐蔽审计收集和将日志信息从源系统转移到日志收集服务器的机制。Sebek2 可以从上面提及的 Honeynet Project 工具页面获得。

Sebek2 可以收集更多的信息，包括应用程序中的击键以及键入的密码。在本章中，更令

我们感兴趣的是它的保护措施，它将隐蔽日志记录推向极致（当然是在“白帽 - 黑帽分界线”的这一边）。

下面是 Sebek2 中的一些保护手段（2004/12）：

- 隐藏进程：Sebek2 不修改系统中的任何程序，而是作为一个隐藏的内核模块存在。它也可以编译形式存在，通过编译嵌入到主内核代码中，提供额外的隐藏选项。
- 隐藏内容：不像上面的 bash shell，Sebek2 用 SSL 加密从系统传出的所有数据。
- 嗅探保护：Sebek2 修改内核来阻止数据包被蜜罐嗅探到。因此，即使是少数试图监视本地网络的攻击者，现在也知道自己正在被监视了。

关于 Sebek 架构和使用的更多细节参见白皮书《Know Your Enemy:Sebek》，地址为 http://www.honeynet.org/papers/sebek.pdf。

Sebek 可以用于生产环境中，高度隐蔽地监视用户和潜在入侵者的活动。在生产环境中进行这类隐藏日志记录有多种好处，包括对取证分析（存在许多证据）和信任用户（例如系统管理员）监测的明显益处。当审计很可能遭到信任的攻击者破坏时，信任用户监测这一项功能将变得非常方便。

另一方面，sebek 的主要缺点是日志数据的容量巨大。这种日志记录方式比大部分标准审计系统机制更详细、粒度也更细。

6.4 隐蔽日志通道简述

从 Sebek 更进一步，我们就进入了隐蔽通道的领域，这是一个激动人心的领域，但是可能超出了本书的范围。我们将简单地讨论现代隐蔽通道和它们在日志记录中的应用。

“隐蔽通道”（在来自 NSA Rainbow 序列的“Light Pink Book”中第一次正式提出）的经典概念现在几乎已经废弃。更现代的隐蔽通道实例处理使用基本 TCP/IP 协议及常见应用协议的信息传输。当然，也可以使用更底层的协议（如之前提到的 CDP）。

例如，TCP（UDP 也是）和 IP 报头的许多字段没有定义（例如服务类型（TOS）/ 显式拥塞通知（ECN）TOS/ECN 字段）、未设置（填充）、设置成随机值（如初始序列号）、设置成多变值（IP ID）或者可选（如选项）。这个非常重要的事实造成了在不破坏 TCP/IP 标准（从而丢失数据包）和使数据包表现异常（从而触发检测系统）的情况下，隐蔽地增加传送信息的可能性。而且，许多更高层的协议（如 HTTP）可以配置成携带数据（例如通过 HTTP 隧道工具），用于通过一系列 web 访问序列，建立一个 shell 会话通道。

更特别的是，有一些工具可以通过思科发现协议（Cisco Discovery Protocol，CDP）来传输数据（可能是日志数据）。CDP 是一个用于网络设备通信的非路由协议。

什么场景下会选择使用这种程度的日志记录隐身？蜜罐网络或高安全性环境中受到严密监控的生产服务器就是两种可能的场景。

6.5 小结

本章介绍了隐蔽的日志记录，它可以用来加强日志记录基础设施的保护，补充其他安全措施（避免“隐藏安全性”）。

本章的要点如下：

- 隐藏日志记录工具可用于有益的目。
- 蜜罐是隐藏日志记录常用的环境。
- 隐藏日志记录最终是可以发现的，这就是它不能作为唯一保护手段的原因。

参考文献

A guide to understanding covert channel analysis of trusted systems (ncsc-tg-030 version-1, "Light Pink Book") <http://www.fas.org/irp/nsa/rainbow/tg030.htm>, National Computer Security Center, November 1993.

Bauer, M. (2002). Stealthful sniffing, intrusion detection and logging. <http://www.linuxjournal.com/article/6222>.

Honeynet Project (2004). *Know Your Enemy*. (2nd ed.). AWL.

Know Your Enemy: Sebek <http://www.honeynet.org/papers/sebek.pdf>.

Spitzner, L. (xxxx). Honeypots: tracking hackers.

Chapter 7

第 7 章

分析日志的目标、规划和准备

本章要点

- 目标
- 规划
- 准备

7.1 概述

迄今为止，我们已经介绍了日志数据的概念、日志数据架构、集中化日志记录、日志留存的基本知识，以及一些其他的主题。现在是投身日志分析的时候了。但是在我们开始讨论技术和方法之前，我们必须打下一些基础，了解实际开始分析日志数据之前需要做的工作。本章将介绍三个重要的主题：目标、规划和准备。顾名思义，日志分析目标是你希望通过日志分析实现的目标。规划是进入日志分析过程前的第二步，我们规划日志分析的方法。最后，我们将介绍准备。这涉及日志数据的准备，和处理日志分析活动的环境准备。

7.2 目标

根据你的特殊需要，分析的目标会有所不同。如果你所处的是一家银行机构，目标和连锁饭店就会不同。但是，有一组较高级别的目标可以适用于大部分人。

7.2.1　过去的问题

日志分析的主要目标是了解已经发生的事情，并且能够对它们保持警觉。本节提供一些“过去的问题”的例子。

创建新用户（Windows）：

```
EvntSLog:423: [AUS] Fri Oct 05 11:59:09 2001: HANDCUFFS/Security
   (624) - "User Account Created: New Account Name: tbird New Domain:
   HANDCUFFS New Account ID: S-1-5-21-1647595427-22557637-1039276024-
   1001 Caller User Name: Administrator Caller Domain: HANDCUFFS Caller
   Logon ID: (0x0,0x2B79) Privileges- "
EvntSLog:424: [AUS] Fri Oct 05 11:59:09 2001: HANDCUFFS/Security
   (626) - "User Account Enabled: Target Account Name: tbird Target
   Domain: HANDCUFFS Target Account ID: S-1-5-21-1647595427-22557637-
   1039276024-1001 Caller User Name: Administrator Caller Domain:
   HANDCUFFS Caller Logon ID: (0x0,0x2B79) "
EvntSLog:425: [AUS] Fri Oct 05 11:59:09 2001: HANDCUFFS/Security (628)
   - "User Account password set: Target Account Name: tbird Target
   Domain: HANDCUFFS Target Account ID: S-1-5-21-1647595427-22557637-
   1039276024-1001 Caller User Name: Administrator Caller Domain:
   HANDCUFFS Caller Logon ID: (0x0,0x2B79) "
```

SSH CRS-32 攻击：

```
sshd[6169]: fatal: Local: Corrupted check bytes on input.
sshd[6253]: fatal: Local: crc32 compensation attack: network attack
   detected
```

sendmail 攻击：

```
Jul 21 01:25:49 ariel sendmail[308]: BAA00307: to=lamarbroc@delphi.
   com, ctladdr=":/bin/mail root@ariel.sdsc.edu </etc/passwd",
   delay=00:00:34, mailer=smtp, relay=bos1h.delphi.com. (192.80.63.8),
   stat=Sent (Ok.)
Jul 21 01:35:40 ariel sendmail[545]: setsender: "/bin/mail root@
   ariel.sdsc.edu </etc/passwd": invalid or unparseable, received from
   [205.133.101.5]
Jul 21 13:13:04 ariel sendmail[784]: NAA00783: to=\tsutomu,
   ctladdr=tsutomu@ariel.sdsc.edu, delay=00:03:09, mailer=local,
   stat=Sent
```

Cisco IOS 更改：

```
%SYS-5-CONFIG: Configured from host1-config by rcp from 172.16.101.101
```

FW-1 MAD 端口扫描：

```
Feb 26 17:47:28 host.example.com root: 26Feb2001 17:47:28 accept
   localhost >daemon useralert product MAD proto ip src elendil dst
   host additionals: attack=blocked_connection_port_scanning
```

第 9 章将介绍如何使用关联，轻松地应对这些问题。

7.2.2 未来的问题

除了发现已知的问题之外，你还应该检测未知、之前从未见过或者已知范围之外的事件。为什么这样做很重要？因为攻击者会发展自己的技术，用通常的方法去检测不正常的行为（例如查看日志数据以发现你所知道的情况）最终将变成攻击者进入你的环境的途径。这种分析通常不容易通过人工审核日志实现，而需要更加复杂的技术。第 11 章将讨论数据挖掘技术的使用，而第 10 章说明如何使用统计分析找出你从未见过的情况。

7.3 规划

日志分析系统的需求和你的目标一样关键。如果你不理解规划分析系统的关键概念，你就会遭到某种程度的失败。本节关注系统规划中较高级别的考虑因素。在商业和开源解决方案、托管安全提供商以及你自己构建的系统上进行日志记录时，这些概念将成为可以引用的蓝图。

7.3.1 准确性

准确性是任何日志记录系统的基础。但是，它在日志记录系统中有何含义呢？坦率地说，准确性就是确保接收、处理和存档的日志数据没有缺陷或者误导性信息。换言之，接收到的数据正是你或者日志记录系统所希望的形式。假阳性是这方面的问题之一。入侵检测系统（IDS）就是假阳性的来源之一。例如，IDS 可能报告恶意的流量，而实际上该流量不是恶意的，或者不可能代表着目标系统上的真正漏洞。对于 IDS 来说，解决方案之一是查询一个漏洞数据库，交叉参考可能的恶意行为。进一步，你的 IDS 还可以实现一个策略方案，使用用户和组配置文件来创建个人可接受的网络使用模式。

减少假阳性有助于将混乱现象降到最低限度。显然，如果你消费来自供应商的日志数据，几乎没有办法控制其传输的内容。但是如果你计划编写一个定制的日志记录系统，一定要记住，和人们的交谈一样，保持内容简明扼要往往很有用。

关于准确性的最后一个领域涉及时间戳的概念。时间戳是有些事件发生的日期和时间。地球上的几乎每个安全应用程序都使用时间戳。而且，大部分这些应用程序在转发任何日志消息时也发送时间戳。问题是不同的供应商使用不同的格式，有的使用 UNIX 纪年方式，而其他可能使用标准格式化的日期和时间字符串，更糟糕的是使用自己的格式。与时间戳相关的主要标准是 ISO8601 标准。下面是 ISO8601 时间戳的一个例子：

2004-07-21T11:44:44+00:00

例中的格式是 YYYY-MM-DDTHH:MM:SS.SSS+/－H。这里的字母“T”是为了分隔日期和时间。+/－是与格林威治标准时的偏差。此外，RFC 3339 定义了用于互联网的日期和时间格式。

最后，下面是各种系统的日志时间戳的一些例子：

- Fri Aug 31 15:44:13 2007
- 10/Jul/2006:00:02:20 −0400
- Jan 11 10:36:21
- 10/21/2005 11:11:38
- 2004-08-06 10:32:53
- 12/2/2003,3:29:05 PM
- 07-06-200400:00:49
- <time offset="0" timeZone="GMT">1041880724715825000</time>

可以看到，日志时间戳格式是个真正的问题。遗憾的是，你很有可能必须处理这样的时间戳格式。

7.3.2　完整性

完整性涉及来源的信任，有些时候还涉及日志消息内容的信任。如果你依赖商业日志数据，那么你对供应商就无计可施。有些供应商了解这一点。例如，有些商业化产品不仅在客户端和服务器身份认证中使用安全套接字层（SSL），而且还将其用于数据本身的加密。其他供应商依赖于旧的 syslog 或者 SNMP。诚然，最新版本的 SNMP（版本 3）考虑了安全和加密，但是许多供应商还不支持。如果你依赖于供应商并不安全的日志数据，可以使用 STunnel（http://www.stunnel.log）等工具。STunnel 是一个通用应用程序（以 C 语言编写），允许你通过 SSL 隧道转发和接收任何类型的数据。当你需要安全通信，但是对数据本身没有控制时，或者没有时间了解特殊的 SSL API 时，这个工具就很方便。

另一个解决方案是以明文发送数据，但是使用能够避开窃听和篡改的专用网络链接。但是，这是一种昂贵的方案，因为你可能已经有了多用途的网络链接。明显，你无法使用这些链接进行敏感数据的传输。

关于完整性的最后一部分信息涉及数字签名。如果你对日志数据进行数字签名，就可以确保它可信。如果日志数据最终用于犯罪调查的证据，有些监察部门可能要求在你的系统中实现这种级别的完整性。

7.3.3　可信性

我们想起了电影《War Games》中的场景，在乔舒亚试图说服北美防空司令部（NORAD）核破坏很快会来临时，某个控制台上的一位操作人员通过喇叭通知这个消息的可信性很高。可信性与你对已经发生、将要发生或者值得关心的事件的确定性有关。有些产品试图在给定事件上指明优先级或者严重性来做到这一点。这种概念性的想法可能采用“高”、“中”、“低”的形式，或者是一个从 0 至 99 的比例。例如，许多 Cisco 设备使用了 0 至 7 的级别。

实际中的可信性比简单的优先级方案更深入。真正的可信性来自于对网络和基础设施能

力的了解。SIEM 系统对此很有用，这一市场相当新颖且不断发展。SIEM 是典型的企业应用程序，建立在网络中所有安全设备的基础上，包括但不限于防火墙、IDS、漏洞扫描器、数据库和应用程序。所有系统为 SIEM 提供内容，SIEM 可以使用统计算法分析日志数据，或者允许一位操作人员创建自定义规则，查询日志消息中的特定模式。创建自定义规则的好处是可以抽象潜在的安全威胁，根据同质的日志数据捕获这些威胁。(例如防火墙和 IDS)。这些操作处于可信性的核心：从许多领域（通常是不相关的）获取输入，并从所有输入中衍生出更为成熟（和准确）的事实。

7.3.4 保管

保管的思路是日志数据不应该以任何形式被修改。如果你计划使用日志数据对恶意操作提起诉讼，这是必备的条件。例如，许多安全设备供应商提供了自己的潜在攻击类型概念，例如防火墙断线、IDS 端口扫描等。在日志系统接收、处理、分析和存档过程中的任何时点，这些信息不被更改都是至关重要的。

有些供应商在事件分类上做得很糟糕，这一再引起争议，有些聪明的网络安全工程师可能觉得自己知道得更多，希望将供应商提供的事件类型描述改为工程师认为更能表现“实际”入侵、威胁等现象的描述。动机是好的，但方法欠妥。在这种场景下，工程师应该拥有一个日志系统，在独立于原始数据的情况下，进行这种映射。

7.3.5 清理

在处理日志系统时，日志数据的可信性有时可能是一个关注点。例如，你可能有不想通过网络链路传输的用户账户信息或者 IP 地址，或者你可能打算向第三方供应商传输日志并且希望保持某些内容的私密性。日志清理有两种主要技术。第一种是检测和替换或者删除你想要清理的条目。例如，IP 地址 10.0.3.4 可能变成 xxx.xxx.xxx.xxx。这会警告查看清理后日志条目的人，日志文件中有一个 IP 地址。另一种技术基于这样的方案：删除的日志数据可以在以后重建。例如，清理的信息可以提取出来放到一个安全的文件中。

7.3.6 规范化

日志和分析系统中的规范化是一个有趣的主题。我们现在关心的是将供应商数据规范化为安全和网络分析人员所能操纵、报告的格式。换言之，我们关心的是创建一种众所周知的日志事件格式。许多安全供应商选择自己的日志记录格式，而不理会使用的是 syslog 还是 SNMP。更糟糕的是，有些供应商选择提供专有的软件开发包（SDK）来访问应用程序的内部数据格式。在你真正想要处理日志消息，提取 IP 地址和端口信息等数据时，这可能成为真正的问题。规范化有时被称作统一，实质上是将许多不同供应商的日志数据格式映射到我们了解的格式——也就是你已经有能力处理的格式。

正如我们在第 4 章中所看到的，为存储数据定义一个逻辑模式是规范化拼图的一部分。通过提出符合应用程序或者组织需求的数据模式，你可以轻松地映射几乎所有供应商日志事件数据，不管它们有多么冗长或者庞大。这使得取证挖掘和历史搜索等工作变得很轻松，因为你预先知道了数据的格式。第 9 章将展开这个主题，讨论解析和规范化。

7.3.7　时间的挑战

规划部分的最后一个主题是时间同步问题。时间是事后调查中非常关键的部分。日志可以确定地表明发生了某件事。下面是时间和日志数据中的一些挑战：

- 日志中完全虚假的时间戳——电池耗尽或者其他硬件故障是出现与实际情况毫无联系时间戳的常见原因。
- 总有某个地方是下午 5 点：你的日志在哪个时区？许多日志文件在日志中没有包含时区；人们需要负责记下日志记录系统所在的时区。
- 你是否正在漂移？你的时钟可能如此——必须承认，NTP 时钟漂移会导致几秒钟的误差，但是即使如此也可能改变事件的顺序（例如，先开始攻击还是先试图登录？由此可以看出登录方式）。在所有系统上使用 UTC 时间可能解决这个问题。
- syslog 转发者的谜团：他的时间和我的时间之间的差别，因为人们采用缓冲 syslog 服务器（例如 syslog-ng 或者能够存储转发日志的商业化系统），这在未来不太可能发生了。在这种情况下，人们必须确保 syslog 消息包含事件的时间，而不是延迟的传输发生的时间。
- 那么，哪个是正确的？系统有两个时间戳！有些系统有不止一个时间戳，其中一个有更高的确定性，代表着事件的时间。
- 如果某个事件的日志时间为早上 5 点 17 分，发生的时间是？日志时滞是真实存在的，特别是对于记录进程或者程序执行的日志。例如，有些系统在程序终止或者退出时记录进程执行日志，而不是在启动的时候。
- 日志中是 5 点 17 分，令人伤心的是，可能意味着上午 5 点 17 分，也可能意味着下午 5 点 17 分（17:17）。有些旧系统会犯这样的错误。
- 最后，你能够信任 NTP 时间源吗？你和时间提供者之间有没有 SLA？这些问题的答案可能是“否”。这确实是非常深奥的问题，但是至少必须略微考虑一下。

下面是一些需要遵守的经验法则，能够缓解这些挑战。

- 信任 NTP；运行一个时间服务器，与有保证的时间服务器（例如 time.nist.gov）同步。
- 实现所有日志记录系统与上述 NTP 服务器的时间同步。
- 在接收和集中日志的时候记下时区（确保使用 EST、PDT 或者 GMT+/-N 格式）。
- 如果定时机制失效，你最终回到 1970 年 1 月，你可以尝试使用其他日志进行关联，不过，在法庭上证明这种定时规程，将是一场困难的战役。
- 为了应付上面提到的日志时滞——了解记录日志的系统！

- 为了得到更高级别的保证，研究信任的时间协议（参阅 ANSI X9.95:2005）。

7.4 准备

本章的最后一个主题是准备。准备正如换挡准备下车一样。我们要确保已经准备了环境和日志数据，不仅能够分析它，还能存储它、利用它做出报告等等。

7.4.1 分解日志消息

如果你能够先根据主机，然后根据服务分解日志消息，分析就会更加容易。它还能使如下的任务更加简单：

- 开发一组独特的模式；如果你按照服务分解消息，就更容易做到。
- 确认每种模式中的“变量”：类型和范围。
- 检查假阳性（用于模式匹配）和损坏的数据。
- 存储模式：可以用扁平的文本文件、数据库等。

7.4.2 解析

解析是获取原始日志消息，并从中提取特征信息。这些特征包括日志消息来源、日志发生的时间戳、源和目标 IP 信息、用户信息等。这使你能够更好地理解你的日志数据，因为你可以使用日志消息的各个部分进行详细的分析和报告等工作。第 9 章更加详细地介绍解析和相关的概念。

7.4.3 数据精简

理解数据精简技术很重要，它不仅能够高效地从源传输到中间处理服务器，还能够高效地存储数据。但是有些技术是有代价的。有些精简技术影响源主机系统、接收主机系统（或者两者皆是）的 CPU 性能。从这些技术上得到的好处有时候超过对系统的影响。如果你能够在网络上发送较少的数据（也就是占用的带宽较少），在通过网络发送的数据有限制的情况下也是一种收获。

1. 数据压缩

数据压缩将一些字节压缩为更小的字节集合，从而使传输该字节串所消耗的带宽更小，存储消耗的磁盘空间更少。然后，压缩后的字符串可以由接收方或者应用程序解压。缺点是在数据被压缩、然后在另一端解压缩的时候，对发送方和接收方都有性能影响。

在日志分析中应用压缩有两种方式，一种是在我们从日志聚合器将规范化的事件传输到处理服务器以供分析和存档时进行。另一种方式是在数据库级别使用压缩，这通常是特定数据库供应商支持的某种功能。

2. 消除重复数据

在日志数据收集的正常过程中，常常会接收到重复的事件数据。从技术上讲，重复事件数据的概念在安全日志数据中几乎从不适用。你最终得到的是有不同特性的同一种事件。例如，假定一个攻击者对你的网络上的主机进行端口扫描。在每次探查中，你的 IDS 都很有可能创建单独的事件。事件类型可能相同（端口扫描或者类似的类型），但是攻击者的源 IP 地址和端口可能在每个事件中都不同，特别是在采用了欺骗技术的情况下，但是目标 IP 地址（被攻击主机）在每个事件中都相同。

安全日志事件分为两种主要类别：信息性的事件和真正具备操作和战术价值的事件。信息性的消息是系统启动或者关闭之类的事件。操作性或者战术性的日志数据是 IDS、防火墙等报告的事件。例如，失败的 SSH 登录尝试就有特别的操作意义。

日志分析的目标是为事件设定唯一的标识符，将看似不同的事件合乎逻辑地结合为单一事件，仅用事件发生数量的计数来区分。这使得发生的成百上千个事件可用单一事件代表，节约了大量空间。

日志数据中有几部分信息可以帮助你进行重复消除：

- 日志事件来源（IDS、防火墙等）
- 源 IP 地址
- 源端口
- 目标 IP 地址
- 目标端口

日志事件来源：日志事件的来源是生成事件的设备、软件等，有多种含义。许多安全产品转向管理服务器架构。这意味着，每个设备不再向日志服务器单独报告，而是向一个管理系统报告，该系统代替一个或者多个设备，将事件转发到日志服务器。这对于将日志事件传输到第三方接收者的供应商来说是更为高效和可伸缩的方式。缺点是，不管使用 SNMP 还是 syslog，日志事件的物理来源是管理服务器，也就是说，我们在 syslog 消息中将看到管理服务器的 IP 地址。为了识别实际生成事件的传感器，管理服务器将在它发向日志服务器的日志消息中放置事件来源。注意，我们使用“传感器”这一通称：传感器可能是防火墙、IDS 等。现在，正如没有进行标准化的事件日志格式，来源字段的格式也是供应商特定的。它可能是传感器的 IP 地址或者主机名，也可能是一些独特的标识符。当你打算使用日志事件来源作为消除重复数据的唯一关键字时，应该意识到这一事实。

源 IP 地址：源 IP 地址是攻击的发源地。在实践中，这个地址不足以保证唯一性。这主要是因为攻击者可能伪造一个或者多个源 IP 地址。

源端口：源端口是来源系统随机创建的端口。同样，由于欺骗的存在，这在消除重复中通常是不可靠的。

目标 IP 地址：目标 IP 地址是目标系统的 IP，通常在攻击期间保持不变。这是有助于消

除重复数据工作的合适候选。

目标端口：发生端口扫描攻击时，IDS 将报告同一个目标 IP 地址的许多不同端口。对于许多攻击者来说，只有一个目标 IP 地址和一个目标端口。目标端口也是消除重复数据过程的一个合适候选。

3. 非工作时间数据传输

从技术上讲，这个主题不是一种数据精简技术，但是在本章中将简单地加以介绍。这种技术对于从聚合器传输到主 / 分析服务器上很有效。基本的原则是确定每天中的某个时间来传输你不一定希望分析、但是希望存档的数据。例如，如果你部署了非常严格的防火墙该规则，你可能不关心防火墙接受、允许等事件的查看和分析，但是你仍然需要存档所有这些事件。利用非工作时间传输，你可以在网络流量最小的某个时点（比如凌晨 3 点）发送这些防火墙事件。在你的服务器上不对这些事件进行分析，还能降低负载，将 CPU 周期留给更急需处理的事件。

实现这种技术的步骤相当简单。首先，你需要一种方法，来检测和隔离希望在以后传输的日志消息。这可以通过使用 syslog-ng 的内建机制，或者系统支持的类似机制来完成。这些消息必须被保留在日志聚合点，但是存储在不会影响更高优先级事件传递的位置。其次，必须定义事件传输的日期和时间。最后，在指定时间将事件传输到主服务器进行分析和存档。你还需要一种手段，告诉服务器这些事件不需要分析；如果你在较晚的时间传输这些事件，处理它们就不总是有意义了。应该注意的是，如果你的特殊日志分析工具不支持这个概念，你可能必须用 scp、rsync 或者其他工具创建自己的解决方案。

需要了解的一个问题是，如果你允许指定时间范围（如凌晨 3 点到 4 点），可能会碰到一个小时不足以传输所有隔离事件数据的情况。现在，你必须做出决定：停止发送事件，或者不管时间，继续发送事件直到所有事件发送完成？这取决于你的特殊需求，或者由你的直觉来决定哪个选择正确。

7.5 小结

开始日志分析之前，你必须确保一切就绪。这包括设定目标、规划日志分析系统，以及准备供分析的日志数据。本书剩下的章节将更详细地说明本章中介绍的许多概念。

第 8 章 Chapter 8

简单分析技术

本章要点

- 一行接一行：绝望之路
- 简单日志查看器
- 人工日志审核的局限性
- 对分析结果做出响应
- 示例

8.1 概述

人工阅读日志应该保留，用作地狱中的惩罚。但是，在缺乏好的自动化工具时，人们总是不得不进行人工审核，因此我们在本章中研究这项工作。坚实的基础会让你事半功倍。此外，我们将深入介绍对日志分析结果的响应。

但是，为什么我们要进行日志分析呢？下面是一些原因：

- 依从性和监管的要求
- 了解网络条件
- 基础设施 ROI
- 安全度量
- 事故响应

8.2 一行接一行：绝望之路

在我们开始人工分析的主题时，让我们尝试审查一个用于 Checkpoint Firewall-1 的 Nokia IPSO 平台（BSD Unix 的一个变种）的日志文件。我们将和刚入门的系统管理员或者安全分析人员一样逐行阅读日志，得到这一过程的感性认识：

```
Mar 25 15:56:42 cp41 [LOG_NOTICE] xpand[370]: Sending trap request
   (to default addresses)..
Mar 25 15:56:42 cp41 [LOG_NOTICE] xpand[370]: Trap sent
Mar 25 15:56:42 cp41 [LOG_NOTICE] xpand[370]: Configuration changed
   from 172.16.10.55 by user admin for page /opt/syswatch/templates/
   configuration
Mar 25 15:57:00 cp41 [LOG_INFO] /USR/SBIN/CRON[18212]: (root) CMD
   (/etc/hourly 2>&1 >>/var/log/hourly)
Mar 25 15:58:11 cp41 [LOG_NOTICE] xpand[370]: Configuration changed
   from localhost by user admin
Mar 25 15:58:24 cp41 [LOG_INFO] sshd-x[16837]: Generating new 768 bit
   RSA key.
Mar 25 15:58:25 cp41 [LOG_INFO] sshd-x[16837]: RSA key generation
   complete.
Mar 25 15:59:05 cp41 [LOG_INFO] sshd-x[18383]: Accepted publickey for
   ROOT from 172.16.10.55 port 36710 ssh2
```

现在，我们快速分析这个日志文件：

1）好，第一行表示发送一个 SNMP 陷阱，看上去完全正常。

2）第二行也一样，看上去是陷阱已发送的成功状态。

3）下一行看起来像是来自 IP 地址 172.16.10.55 的管理员做了某些配置更改。这可能是正常的，也可能不正常，取决于我们了解的其他细节。

4）下一行表示从 cron 运行了一个进程，cron 是一个在后台定期运行程序的机制。这看起来是正常的。

5）第 5 行也是一个配置更改，但是没有归因于特定的用户，无法知道它是否重要，但是看起来 IPSO 将它视为低优先级事件。

6）下两条消息与 sshd 远程访问守护进程的例行活动相关，该进程每隔预先定义的时间周期就会重新生成加密密钥，可以忽略。

7）上述日志中的最后一行表示，“root”用户从发起上述修改的同一个 IP 进行连接。这可能正常，除非前面的配置更改本身是恶意的。

你是否感到沮丧？我们只不过分析了特定计算机系统大约 3 分钟的活动；只有 8 行日志、762 个字节。现在想象一下，如果我们从这个繁忙的系统中得到一个月的日志是什么情况。我们将得到由有数百万行代码所组成的大小为几 GB 的日志文件。突然间，人工审核似乎成了一个极其麻烦的过程。确实，说得好听一点，人工日志审核不具备伸缩性。但是，我们在

某些情况下仍然不得不使用它，例如在缺乏工具或者需要查看某些非常特殊的情况时。

另一个难题是，我们要熟悉这种日志格式，对 Unix、IPSO 等环境有总体的了解。分析人员往往必须从头开始学习所有的知识。正如我们在前几章所提出的，二进制和其他不可读格式也会显著地增加复杂度。因此，日志可能难以阅读和理解，这给人工分析造成了障碍。

8.3　简单日志查看器

下面我们来了解一些 Unix 和 Windows 中的简单查看器，以及它们所能提供的帮助。

8.3.1　实时审核

在大部分 Unix 和 Linux 变种上，有多种工具能够辅助日志分析。有些在实时日志查看上有所帮助，而其他则有助于审核历史日志。

我们将从经典的工具开始：

```
# tail -f /var/log/messages
```

上述命令将显示标准 Unix 消息文件的最后几行，以及命令启动之后出现的新行。这是目前最简单的实时日志查看器，但是在大部分情况下没什么用处。不过，如果你所需要的只是在系统日志记录出现时看到它们（例如在新程序测试或者守护进程崩溃时），这个工具就相当方便。

上述命令可以用标准的 Unix 字符串匹配工具 grep 加以改进：

```
# tail -f /var/log/messages | grep sshd
```

上述命令只显示安全 shell 守护进程生成的消息，如：

```
Dec 12 17:31:56 ns1 sshd[4095]: refused connect from 80.92.67.68
Dec 12 17:37:34 ns1 sshd[484]: Generating new 768 bit RSA key.
Dec 12 17:37:34 ns1 sshd[484]: RSA key generation complete.
```

为了进一步改进上述的实时日志查看，我们可以投入一个 tee 命令，它将帮助我们查看数据，并发送到一个文件：

```
# tail -f /var/log/messages | grep sshd | tee file-of-latest-sshd-logs.
  txt
```

上述命令显示日志行，并将其记录到“file-of-latestsshd-logs.txt”文件。

less 的粉丝可以使用这个 Unix 命令的等待模式，完成同样的功能：

```
# less /var/log/messages
```

现在，按下 F 使 less 进入“tail”模式，显示最近和新到的数据。

在 Windows 环境中，捆绑的事件查看器可以监控三种日志中新到的记录。我们将在关于历史日志查看器的下一小节中介绍它。

8.3.2 历史日志审核

既然我们提到了 less，就可以开始谈论存储日志的查看了。显然，你所选择的平台上的任何文本编辑器都可以用于查看普通文本日志。但是要注意一点！有些日志文件非常大，前面我们已经提到了这一点。因此，不是所有文本编辑器都能够高效地处理它们。只要试试在 Windows 记事本中加载一个 1GB 文件（不会成功），你就会明白了。其他编辑器能够正常工作，但是操作极慢。所以，你应该使用轻量级的工具，另一方面，它还必须具备搜索和其他有用的功能。

在 Unix 上，more 和 less 工具通常符合上述要求。它们能够处理大文件（GB 数量级），也有搜索功能。遵循“不重复手册的内容”的原则，我们建议读者参考 Unix/Linux 系统上 man 命令的输出。输入 man more 和 man less 就会显示这些信息。有时候，更多详细的帮助可以用 info 命令得到，如：

```
$ info less
```

在 Windows 上，唯一的捆绑工具是事件查看器（Event Viewer）。在这里我们要说明的是，它能够查阅三种标准的 Windows 日志（应用程序、系统和安全），等待新事件并显示之（单击“刷新”），也可以进行过滤和排序。图 8.1 显示了这个查看器。

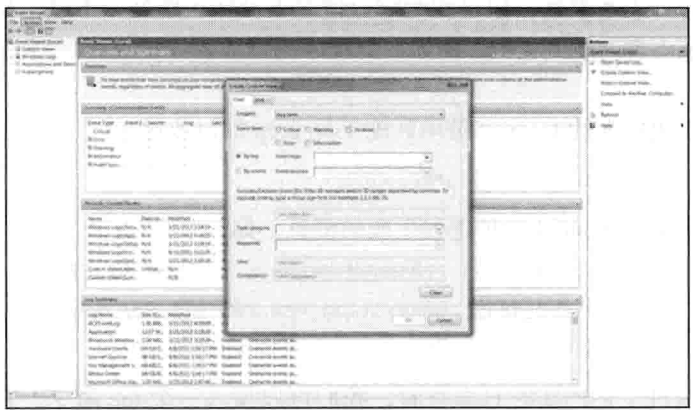


图 8.1　Windows 7 事件日志查看器

如上所述[⊖]，Windows 7 事件查看器较为有趣的功能包括：

- 将单独的事件当作 XML 处理。
- 用事件订阅功能，在远程机器上订阅事件。
- 保存过滤器为自定义视图。
- 在应用程序日志中记录你的自定义事件。
- 运行一个任务，对事件做出响应。

⊖ http://www.razorleaf.com/2009/11/event-viewer-top-5/

上述工具将为你显示日志，但是最有可能的是，你将开始寻求更合适的手段。我们可以进行过滤吗？进行总结？稍微更改一下格式？一些工具能够帮助我们完成这些工作。

8.3.3 简单日志操纵

我们将从 Unix 开始，因为它具备更多这方面的工具，然后我们再转向 Windows。

我们确定，可以在 Unix 上用 tail 和 less 实时查看日志，并用许多文件查看工具（如 cat、more 或者 less）查看存储的文件。

我们首先总结一下使用简单工具的目标，然后说明如何实现这些目标：

1）日志过滤：我们只想看到特定的内容。

2）日志重新格式化：修改我们查看的方式。

3）日志总结：查看紧凑的视图。

如上所述，grep 与 tail 结合可以进行简单的过滤。Grep 还有助于在大的日志文件上进行更为高级的过滤。下面是用 grep 进行常见日志过滤的几个例子：

```
# grep -ev 'ssh|telnet' /var/log/messages
```

查看除了包含 ssh 和 telnet 之外的所有消息。

```
| # grep -f patterns /var/log/messages
```

查看匹配文件“patterns”中模式的所有消息。

改进 grep 的其他工具是 tail 和 head。它们用于查看日志的开头部分和结束部分。例如：

```
# tail - 1000 /var/log/messages | grep ailed
```

上述语句搜索消息文件最后 1000 行中带有字符串“ailed”的记录（目标是捕捉“Failed”和“failed”，因为 grep 是区分大小写的）。类似地：

```
# head - 1000 /var/log/messages | grep ailed
```

搜索消息文件前 1000 行中带有字符串“ailed”的记录。

如果你结合 awk 和 sort 工具，就可以对日志进行更多的处理。例如，我们可以查看文件中记录了哪些设备和系统：

```
# cat messages | awk '{print $4}' | sort -u
```

在我们的测试系统上，上述命令输出如下：

```
ns1
ns2
ns-backup
```

输出的是该服务器上记录日志的 3 个 Unix 系统的名称，我们从一大堆日志中提取出这一摘要信息。本例为我们提供了日志的排序和简单的摘要。实际上，用 awk 能够编写（在功能上和全面性上）复杂的脚本来处理日志文件，但是我们只关注在这里使用的命令。

Log Parser 和 Log Parser Lizard：筛查 Windows 日志的更好手段

在 Windows 上，事件查看器可以过滤、排序日志和将日志提取到文件中。

在典型的 Windows 系统上，过滤限于：

- 事件类型
- 事件分类
- 事件 ID
- 源计算机
- 源应用程序或者系统组件名称
- 用户名称
- 日期和事件范围

为了进行 Windows 日志的更高级分析，我们可以使用具有 Log Lizard 的 Log Parser。Log Parser 是一个 Microsoft Windows 插件程序，为你提供一个类似 SQL 的 Windows 事件日志接口。例如，下面这个查询用于检索 Windows 系统事件日志中的所有行：

```
SELECT * FROM System
```

Log Parser 本身是一个命令行工具，在分析工作中难以使用，这就是 Log Lizard 的用武之地。它围绕 Log Parser 建立了一个美观的界面。这两个工具的安装非常简单直接。你可以从如下位置获取它们：

1）Log Parser:http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=24659.

2）Log Parser Lizard: http://www.lizard-labs.net/log_parser_lizard.aspx.

图 8.2 显示了 Log Parser Lizard GUI。

图 8.2　Log Parser Lizard 屏幕截图

Log Parser 和 Log Parser Lizard 组合的功能包括：

- 图表

- 自定义搜索
- 限制返回的行数
- Via 查询返回结果的导出
- 其他许多功能

Microsoft Log Parser Toolkit 对 Log Parser 提供了很详细的介绍（Giuseppini 和 Burnett，2004）。你可以使用这个参考文献理解它的工作原理以及 via 查询语言的功能。

Log Parser Lizard 有在线文档和论坛，两者都可作为学习这一实用工具的绝佳指南。

8.4 人工日志审核的局限性

到这一阶段，下面这一点应该已经很明显：人工和使用简单的工具和命令进行的日志审核无疑在日志分析中占据一席之地，但是不应该依赖它作为日志分析的主要机制。

人工日志审核有明显的局限性，包括：

- 人工日志审核无法随着日志文件尺寸的增大而伸缩：我们查看了 8 行日志，但是更为现实的任务是查看 800 万行日志。在大规模的企业环境中，日志常常以每秒 20000 条记录的速度产生。在比这低得多的日志记录速度上，人工日志分析就已经无法继续，且变得越来越低效。
- 人工日志审核只能用于简单的日志：当日志文件的意义不言自明，告诉你具体的故障部位时，可以人工解读它，但是在日志来源较为模糊且没有文档时，这种方法必然失败。解读每一行的工作量将从一秒钟的快速思考变成几个小时的在线和其他来源研究。
- 简单的工具和人工审核可能无法获得计算环境中所发生情况的全貌，而更高级的工具能够从日志数据中还原这些情况。
- 除了令人困惑和模糊的格式之外，在许多情况下分析需要关联来自多个源的日志。这种活动确实可以人工进行，但是需要的时间将大大增加（与只查看一个日志文件相比）。这种关联最好留给自动工具完成。
- 最后（但并非不重要），这种活动通常极其乏味。乏味本身就是一个问题，而且通常比来瓶啤酒更容易降低分析人员的效率。

总体来说，只要我们对人工日志审核活动没有太多的要求，它们绝对有自己的位置。我们将在本章最后总结这一结果。

8.5 对分析结果做出响应

人们可能会想，本书是关于日志分析的，当分析完成且确定发生了什么之后，我们就可以停下来了。但是，分析的唯一实际目的是在完成后采取行动。虽然本书的重点明显是分析本身，但是本章将研究一系列行动，这些行动通常根据日志分析进行。特别是，我们将讨论

日志与事故响应的相关性，因为有时候日志会揭示事故已经发生。

到目前为止，我们已经进行了不同的日志分析，是将分析日志的各种原因与现实的行动结合起来的时候了。我们将采取两种级别的行动：微观级别（看到日志某一行或者某个指示时所要做的事情）和宏观级别（作为日志分析结果，必须实施的过程改进）。

8.5.1 根据关键日志采取行动

如果你在网络入侵检测日志中看到如下内容：

```
2004-12-02|00:46:58|dralion1|SSL:COMPROMISE-
SHELL|10.2.3.4|213.190.36.144|443|2328|X||6|tcp,dp=2328,sp=443|
```

然后是：

```
2004-12-02|00:47:02|dralion1|HIPORT:SHELL-
UNAME|10.2.3.4|213.190.36.144|443|2328|X||6|tcp,dp=2328,sp=443|
```

你的行动应该敏捷、果断并且有计划、有准备。为了让不了解这些日志的人们看清楚，我们来做个说明：上述消息来自 Enterasys Dragon 网络入侵检测系统，可靠地表示目标服务器（地址 10.2.3.4）已经遭到攻击者（来自 213.190.36.144）的入侵。注意，在上述的 Dragon 消息中，源和目标地址被颠倒了，因为 NIDS 检测到，从受害系统到攻击者的响应表明，攻击者已经获得了目标 Web 服务器的交互访问权限。为了提供参考，表 8.1 中列出了上述 Dragon 消息的解释。

表 8.1 Dragon 日志消息字段定义

字 段	意 义
2004-12-02	日志事件日期
00:46:58	日志事件时间
dralion1	生成日志的机器的主机名
SSL:COMPROMISE-SHELL	按照 NIDS 供应商分类的攻击名称
10.2.3.4	源 IP
213.190.36.144	目标 IP
443	源端口
2328	目标端口
X|	在本例中没有用到
6	协议（6 代表 TCP）
tcp,dp=2328,sp=443	其他信息

本例的目标是说明，对于某些消息，必要的行动很明显：立刻启动你的事故响应计划。触发不需要太多考虑（或者预先全部考虑过）的立即行动的日志文件和消息是哪些呢？表 8.2 提供了对关键日志采取的行动的一些具体示例。

注意，表 8.2 中的行动不能作为所有情况下的“行动指南”，只是用于说明日志常常代表着实际的行动。

表 8.2　在关键日志消息上所要采取的行动总结

关键日志	必要的行动
可能影响系统操作的故障	如果系统无法操作，需要启动备份系统
成功的攻击	对遭到入侵的机器启动事故响应和恢复过程
很有可能成功的攻击	对遭到入侵的机器启动事故响应和恢复过程
系统达到容量或者某些指标的最大值	增加容量，否则可能导致整个系统失效
可能导致安全和可用性问题的系统变更	如果变更未经授权，撤销更改以避免这些问题
系统崩溃	需要启动备份系统
多次失败的登录	检查攻击者可能猜测密码的入侵迹象
硬件故障	如果系统无法操作，需要启动备份系统
安全相关配置变更	如果未经授权，撤销更改
检测到未授权连接	调查访问系统的人员及其原因

在本节中，我们看到了对日志采取行动的一个简单示例，还没有必要设计和实施一个复杂的计划。此外，我们只提供了行动的一个摘要。例如，在实际工作中，“调查访问系统的人员及其原因”可能演化为一个持续数天的调查过程。

8.5.2　根据非关键日志的摘要采取行动

本小节介绍对非关键日志采取的总体行动。防火墙、路由或者主机消息不会触发像上述 Dragon NIDS 关键消息那样的立即响应。防火墙拒绝一个连接之类的事实通常不会使观察日志的用户发起及时的行动（以近乎实时的方式查看这些日志也是不可能的）。但是，这些消息的组合常常导致间接的行动（否则的话，为什么要分析从来不会导致行动的素材呢？）。

正如 Marcus Ranum 所指出的：“大部分日志文件都是无用的，大部分组织最好是删除这些文件，节省下磁盘空间给真正有效率的内容”（Garfinkel，2005）。这种论调的理由是，如果你从来不查看日志文件（因此也就从不会据此展开行动），保留它们绝对没有意义。虽然在当今这个强调监管依从性的时代，人们会声称“为了依从性的原因”而保留日志，但是上述论据仍然使用，因为如果你和依从性审计人员都不查看日志，并根据在其中找到的迹象采取行动——这些日志就仍然没有用处。从 2011 年的《 Verizion Data Breach Report 》中可以发现，大部分事故在日志消息中都有证据，但是没有人注意到这些日志。这清楚地证明了日志分析确实有益。

我们希望，现在读者已经相信不仅有必要留存日志文件、不仅需要审核和分析它们，还要根据它们采取行动，从中得到最大的价值。

从似乎无法采取行动的日志到真正的行动有许多途径。下面总结一些最常见的途径：

1）根据趋势或摘要采取行动：从摘要中得出发展趋势或者新知识，往往能将许多不能采取行动的消息转化为行动。

2）根据关联事件或者一组事件采取行动：已知的一组非关键事件往往因为其顺序和时间而可以归入关键模式。

3）根据发现的非关键事件模式或者不寻常顺序采取行动：使用高级日志分析工具，人们有时候可能从正常或者无害的日志记录中挖掘出新的可疑模式。

我们来详细地考虑上述 3 种可能性的例子。

- 摘要和趋势是日志分析的常见结果。一个冗长的日志文件可以总结为简短的“10 大攻击”、“最可疑的 IP 地址”或者无限多的其他有用摘要。这些摘要视图往往提示着某种行动。例如，从“占用带宽最多的用户”报告中可能明显地看出，公司排名前三的用户占用了 90% 的可用带宽。这很快会导致一项纪律行动，特别是在这些带宽被用在 P2P 文件共享或者下载与工作无关的材料时。类似地，对路由器 CPU 利用率日志的长期观察，可能揭示不寻常的高活动率时期，导致可能发现攻击者与被入侵系统之间通信的调查。
- 关联（在第 9 章中描述）往往导致一组不重要的事件被标记出来。在这里的一个连接和那里的一个失败登录，以及在其他地方启动的一个应用程序，可能意味着系统入侵或者内部人员对系统权限的违规使用。例如，如果一位管理员在凌晨 3 点访问系统，可能不意味着发生了什么坏事，只是稍微有些可疑。如果管理员接着尝试访问多个应用程序，然后连接到一个数据库并开始加载数据，就可能值得注意了，但是仍然不意味着发生了坏事。但是，如果最后的操作是从同一系统发起的大数据下载，我们就知道很有可能发生了入侵，该对此采取行动了。因此，通过组合一个规则观察这类活动，我们可以根据相对不重要的事件采取（非常必要！）行动。
- 日志中的模式发现是日志分析的新领域（在第 10 章和第 11 章中描述）。它提供了发现日志中的新知识，从而使极其无聊和常规的日志转化为行动的高效和自动化手段。例如，我们可能发现日志文件包含了从一组似乎无关的源 IP 地址发起的重复连接模式。出现这种模式时，可以将其看作新的攻击利用工具在公司网络上的秘密尝试，可能导致安全团队甚至社区全体采取的一系列行动。

表 8.3 总结了行动和无行动价值日志之间关联的更多例子。

表 8.3　与无行动价值日志关联的行动摘要

日志类目 /“记住”	转化为行动的途径	需要的行动
系统状态消息	关联	系统状态消息的某个模式可能表示未授权的系统使用，导致调查和纪律行动
攻击尝试和探查	关联摘要	探查的某种模式可能表示高级攻击者正在寻求突破，必须加固安全防御（这就是一个行动！）
影响较小的攻击	趋势	影响较小的攻击数量的增长可能和更有破坏力的攻击的增长同步，对此要采取行动
系统某些参数接近相对高值	趋势	此类消息的突增表示入侵或者系统总体不稳定，两者都需要采取行动以恢复性能
不同系统的更改	关联	系统更改消息的某种模式可能表示未授权的系统重新配置，导致调查和纪律行动

（续）

日志类目 /“记住”	转化为行动的途径	需要的行动
系统启动 / 关闭	摘要	在某一个时期的大量系统关闭可能代表系统变得不稳定，需要修复
成功登录	关联	在多次失败登录之后的成功登录表示暴力破解密码攻击已经成功
硬件状态消息	摘要	总结各个系统的硬件状态消息，有助于找出问题最多的系统，导致总体优化计划的实施和性能改进
连接建立 / 终止	发现	日志挖掘有助于发现代表新黑客工具的连接模式，导致加固安全防御的需求

表 8.3 说明，如果正确地分析，即使无害和常规的事件也常常需要行动，通过早期预测可能出现的问题，能够避免许多令人头痛的麻烦。

8.5.3　开发行动计划

上述的例子使我们能够归纳出宝贵的“日志分析最后一步”——行动。

这样的计划将包括分析的所有可能结果（以及结果不符合预定条件时采取的行动），覆盖了关键和信息性的警报。

必须承认，这个计划和事故响应计划（见第 16 章）有紧密的关系，但是还将覆盖一些危险的早期征兆（不是事故的组成部分）以及活动常规模式中的变化。典型的事故响应计划包含通信、升级和响应。第 16 章说明如何根据日志数据开发这样的计划，以及根据环境进行调整的深入见解。

8.5.4　自动化的行动

我们介绍了行动的规划和不同类型，但是遗漏了一种潜在的关键行动：自动化行动。如果你可以编程一个系统来自动发起行动，为什么要让人来执行呢？虽然这很有诱惑力，但是实际实施起来会面对大量的挑战。

有哪些可能自动化的行动呢？哪类环境和情况需要实现它们？

在许多关键日志事件中，自动化的行动无疑是合适的。例如，本书的作者之一就曾经在蜜罐网络中使用日志驱动的自动化行动。应该承认，蜜罐环境的重要性低于生产环境，但是风险往往相同甚至更高。蜜罐最为关键的需求之一是始终监控。因此，如果某个监控功能失效，蜜罐的运营也应该停止。这在逻辑上与许多操作系统的功能类似，这种功能就是——操作系统可以配置为在收集的审计停止工作时“优雅地失效”。

因为蜜罐使用多种监控机制，主要的故障场景是磁盘空间耗尽：如果没有空间，任何监控都无法运作和记录攻击者的步骤。因此，需要有一个脚本运行“ df”命令，并观察 /var/log/messages 文件，跟踪“ write failure—check your disk space”（写入失败——检查磁盘空间）、“ n space left on device”（设备上剩余空间为 n）、“ disk full”（磁盘满）或者“ insufficient storage available at the moment”（可用存储不足）等消息。如果 syslog 中出现这些消息或者

“df”命令显示保存日志数据的分区已满，该脚本将运行，通过特殊的防火墙配置请求，阻止对蜜罐网络的所有出入站访问。

8.6 示例

本节将介绍两个根据日志采取行动的例子：紧急事故响应中的日志分析和例行分析结论审核。

8.6.1 事故响应的场景

这个场景处理遭到攻击的公司。事故调查人员已经进行了分析（更恰当地说是“试图分析”，因为日志已经被摧毁）。“Rats R Us”是全球领先的老鼠管理公司，面向企业与个人的端到端管理。该网站为公司贡献了很大的一部分收入，他们已经构建了一个 DMZ 架构，有两个防火墙保护公开的服务器（Web、Email 等）。2005 年 5 月的一个周日，入侵预防系统开始生成大量与公司网络上运行的 IRC 程序（Internet Relay Chat，互联网在线聊天）相关的警报。

从上述日志观察中采取的第一个行动是让一台机器上线，这确实是一个激烈的行动。这提供了根据关键日志采取行动（类似于上面的 Dragon NIDS 警报的例子）的一个例子。事故调查人员尝试搜索日志，但是它们都已经被删除。接下来，分析人员尝试了防火墙的外部连接日志。

正如你从上文中所了解的，这些日志不能立即转化为行动，而需要进行总结、关联或者挖掘。但是，摘要在这种情况下没有什么实际的效果，调查人员也没有任何高级的日志分析工具。防火墙包含了常规的“噪声”，即使可能有一些有趣的事件，发现它们也需要有一个出发点。否则，日志只是一些随机的 IP 地址通过防火墙连接的组合，无法表现出哪一个是攻击者。因此，他们无法根据这样的日志采取行动，事故在大体上没有得到解决。

他们采取的全面和最终行动是规划出一般性的安全建议，包括安装一个安全信息及事件管理（SIEM）产品，以便主动地分析常规的事件日志并据此采取行动。

8.6.2 例行日志审核

现在，我们来看看吸取了事故中发现的所有情况和经验教训之后的同一家公司，他们更加主动地进行日志分析和审核。该公司确保日志配置为发送到一个商业的日志收集和分析产品——SIEM，该产品能够安全地传输和保存日志，并且能够实时和长期地进行高级日志分析。实际上，“Rats R Us”现在已经成了日志记录的天堂。我们再来举个例子，看看他们是如何根据日志、规程和其他信息采取行动。

该公司现在对于根据日志采取行动有了一个清晰的计划。每天，该公司进行各种日志摘要和趋势分析，以观察安全性的“全局”。这些摘要包括公司内的“最常遭到攻击的端口”、

“最常见的事件类型”、“增长最快的事件类型”和“占用带宽最大的用户”。如果观察到某些异常的情况或者任何指标显著增长，都相应地采取行动。同样，如果连接事件的数量下降（“网络连接摘要”报告），就定义另一个行动，因为这可能表示该服务器的性能受到影响。

其他重要性略低但是非常有效的摘要包括“最不常见的事件类型”、“过去没有出现过的事件”和“相关联的事件”。这些都有助于发现日志中少见但却很关键的事件。

8.7　小结

本章试图用暴力对抗日志问题：使用人工日志审核和简单工具。在实践中，这就像在没有攻城武器的情况下，试图用棍棒进攻一个防御严密的要塞：从理论上讲，这是可能做到的，但是需要大量的人力或许多时间（可能两者都需要）。此外，在这一过程中牺牲会许多攻击者，这可不是你希望发生在宝贵的 IT 人员身上的事情。下一章关注更多自动化的分析工具和技术。

本章的要点如下：

1）使用本章中的方法和技术的情况：

- 所需要的就是简略地看看日志（例如快速检查某项情况）。
- 日志体积不太大。
- 分析人员熟悉所要审核的日志格式。
- 所要分析的日志文件包含了严重的关键警报，不需要进一步分析就可以立即行动。

2）不使用本章中的方法和技术的情况：

- 需要了解全局或者趋势（例如高级管理人员所需要的）。
- 日志分析需要高度自动化和智能。
- 日志分散在不同位置。
- 需要从大量分散和非关键的日志中提取一些有行动价值的信息。

参考文献

Garfinkel, S. (2005, April 1). Another look at log files. CSO. Web March 25, 2012. <http://www.csoonline.com/article/220233/another-look-at-log-files>.

Giuseppini, G., & Burnett, M. (2004). *Microsoft log parser toolkit*. Rockland, MA: Syngress.

Verizion 2011 data breach report. Verizion. Web June 1, 2012. <http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2011_en_xg.pdf>.

Chapter 9

第 9 章

过滤、规范化和关联

本章要点

- 过滤
- 规范化
- 关联
- 常见搜索模式
- 未来

9.1 概述

在日志记录和日志管理的世界里，管理员的一大部分时间都花在审核日志、寻找潜在问题上。过去，这一任务大体上由人工进行。当今的日志分析系统提供自动化机制，减轻了这一负担。本章提供对这一过程的深入理解。虽然没有使用特定的产品，但是在过滤、关联、解析、规则等主题上提供了丰富的示例，目标是为你提供必要的概念，不管你使用何种日志记录系统，都可以利用它们。

本章的核心是三个大概念：过滤（filtering）、规范化（normalization）和关联（correlation）。过滤是对原始日志数据采取的行动，确定你是否应该保留它。过滤的输出是规范化的日志数据，这些数据作为关联的输入。关联是为了采取行动而对数据中已规范化的一个部分或者一系列数据进行匹配的工作。图 9.1 说明了这一过程的逻辑流程。

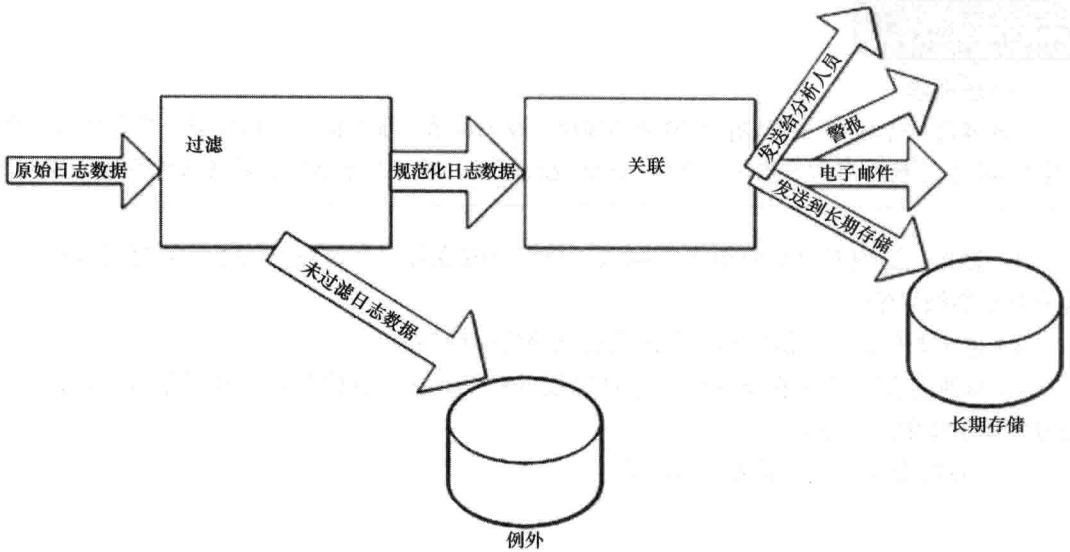


图 9.1　过滤和关联基本流程

我们来简单地描述过程的每个步骤：

1）原始日志数据：这是你的起点，是过程的第一个输入。

2）过滤：在过滤阶段，我们搜索关心和不关心的日志消息。我们不关心的日志消息可以“抛弃”，以减少整个系统的负载。在图 9.1 中，这一步显示为指向一个例外存储的箭头。例外存储可以在以后用来审核较不感兴趣的日志消息。

3）规范化：在这个步骤里，我们获取原始日志数据，并将其各个元素（源和目标 IP 等）映射到一个公共的格式。这对于关联步骤很重要。原始日志消息规范化时，其结果通常被称为“事件”（event）。这个术语在本章中都将被使用，表示一条规范化的日志消息。规范化过程的另一个步骤是分类，也就是将日志消息转换为更有意义的信息块。例如，供应商可能在日志消息中只提供“ID=6856”之类的信息，但是这些字符对于该供应商来说指的是一次登录失败。规范化步骤编写者的工作就是进行这种转换。这一步的额外好处是使得关联规则的编写更加可靠和直观，稍后将介绍更多这方面的知识。

4）关联：关联通常导致当许多组单独出现时将不重要的事件标记出来。这里的一个连接和那里的一次失败登录，再加上其他地方启动一个应用程序，可能意味着系统入侵或者内部人员滥用系统权限。关联的两个基本形式是基于规则和统计。

5）行动：行动通常是发生关联之后你所做的事情。

图 9.1 展示了多种行动（这并不是完整的列表）：

1）发送给分析人员：如果你有一个日志监控界面（通常是某种 GUI），这就是你发送需要立即关注的高优先级事件的地方。

关联的概念

一般概念

本书的作者对这一领域有近 20 年的经验。我们提供一般的概念，以便提供在选择或者使用日志记录系统时所需的基础。当然，我们也会展示一些可用于解决关联需求的实际工具。

2）警报：这通常是向分析人员发送一个事件的变种。在这种场景下，警报可能是一组表示发生高级别情况的事件。

3）电子邮件：这可能用作向值班人员报警的一种手段。

4）长期存储：长期存储是你保存日志数据和规范化事件的场所。这是报告、审计、长期分析等活动的先决条件。

下面我们来看看过滤所涉及的活动。

9.2 过滤

过滤可以归纳为如下操作：

1）保存或者抛弃你所关心的日志数据。

2）将原始日志消息解析为常见的格式，以便更容易地分析数据。

你可能已经知道，供应商不会相互标准化日志消息格式。例如，尽管都使用 syslog 作为底层消息传输机制，Cisco 的日志消息格式与 Palo Alto Networks 或者 Sonicwall 完全不同。第 2 个步骤的强大之处在于能将不同的数据格式转换成一种通用的规范化格式。这对于高效的分析和关联（很快将作介绍）非常关键。但是，你如何知道应该保存哪些、不应该保存哪些呢？我们先来看一个概略的示例。

人为忽略

杰出的 Marcus Ranum 在 1997 年创造了人为忽略（Artificial Ignorance）一词。这个概念的核心是寻找你所熟悉的日志数据，从而找到你尚不知道的事物的一种机制。他提供了如下 UNIX shell 命令，辅助这一过程：（Ranum，1997）。

```
cd /var/log
cat * | \
   sed -e 's/^.*demo//' -e 's/\[[0-9]*\]//' | \
   sort | uniq -c | \
   sort -r -n > /tmp/xx
```

在 sed 命令中，“demo”字符串是运行命令的系统名称。思路是提取这个字符串和日志消息的前导时间戳，以便减少日志数据中的可变性。这是关键！

当你运行这条命令时，将会看到如下的输出：

```
297 cron: (root) CMD (/usr/bin/at)
167 sendmail: alias database /etc/aliases.db out of date
120 ftpd: PORT
61 lpd: restarted
48 kernel: wdpi0: transfer size=2048 intr cmd DRQ
... etc
```

日志消息前面的数字显示该消息在日志文件中出现的次数。当你看到输出的最后时，会注意到数字越来越小。这样做的思路是，因为这些日志消息出现的频率低于其他日志，你可能不了解它们。正如 Ranum 的文章中所说明的，你可以将已知的事物放在一个忽略文件中，以便排除它们。

大部分日志分析系统都提供执行过滤的机制。但是重要的是，宁愿保留更多的数据，也不要随意过滤掉。本节提供的例子只是为了让你熟悉概念，以便了解它，理解它的用途。下一小节和规范化有关，将带我们进入过程的下一步。

9.3　规范化

过滤后的下一步是规范化。我们假定此时你已经知道需要保留的日志数据了。规范化意味着我们取得已知的日志消息，将它们解析为不同的部分（时间戳、IP 地址等），并将它们转换为通用的格式。这种通用格式历史上曾经使用关系型数据库系统（RDBMS）或者其他低级格式（磁盘上的二进制文件等）实现过。但是，在今天的大数据和“No SQL”运动中，越来越多的供应商远离数据库，因为它们不具备可伸缩性。

规范化原始日志消息的步骤如下：

1）获取你所使用的产品的文档。

2）阅读文档中原始日志数据形式及每个字段的说明。

3）提出格式化数据所用的对应解析表达式。大部分日志分析系统利用正则表达式来解析数据。

4）在样本原始日志数据上测试解析逻辑。

5）部署解析逻辑。

不管规范化事件所使用的最终存储机制是什么，某些字段都比其他字段更常用、更实用。这些字段包括：

- 源和目标 IP 地址：这在关联过程中非常有用。
- 源和目标端口：这些数据用于理解哪些服务试图访问或者被访问。
- 分类学：分类学是分类和编码日志消息含义的一种手段。例如，所有设备供应商都生成某种日志消息。这些消息通常映射到登录成功、失败、尝试等。登录成功的一种

分类学示例可能是：Login.success。分类学的重要性在于它使你可以组合类似的消息（从分析的角度），而不需关心生成日志消息的特定供应商。

- 时间戳：在日志数据的世界里，我们通常关心两类时间戳：日志消息在设备上生成的时间和日志记录系统接收日志消息的时间。
- 用户信息：如果提供了这个信息，捕获它们（用户名、命令、目录位置等）往往是很好的事情。
- 优先级：有些日志消息自身包含了某种优先级。这显然是供应商对日志消息优先级的评估，但是可能和你的实际想法不相符。所以，作为规范化工作的一部分，你必须理解特定日志消息对你的环境的影响。典型的优先级值是低、中、高。
- 原始日志：作为规范化过程的一部分，你应该保留原始日志数据。这用于确保规范化事件的有效性。另一个用例是日志留存。你可能有在一段时间内保留原始日志数据的需求。对此有两种解决方案。你可以保存原始日志，作为规范化事件的一部分，或者将其保存在磁盘上，提供一种从规范化事件中“取回”原始日志消息的手段。

我们来看一些有助于理解概念的规范化示例。

9.3.1 IP 地址验证

发现 IP 地址往往很重要。我们已经多次看到如下正则表达式的使用：

```
\d+\.\d+\.\d+\.\d+
```

虽然这能够捕获 IP 地址（如 10.0.3.1），但是也会捕获到无效的 IP 地址（如 300.500.27.900）。提出一个匹配 IP 地址的正则表达式的关键是不仅要检测到四个用句点隔开的数字，还要确保每个八进制数在正确的范围内。下面的正则表达式将验证 IP 地址：

```
^([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\. ([01]?\
   d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])$
```

注意，该表达式被分为两行，但是，它应该在同一行上。还要注意，这个表达式将会检测 IP 地址 0.0.0.0，某些网络类型会认为它是无效的。但是，某些安全系统将伪造的 IP 地址报告为 0.0.0.0，所以检测这个 IP 在某些情况下是有效的。

9.3.2 Snort

Snort（www.snort.org）是一个开源入侵检测系统。下面是 Snort 生成的 syslog 消息示例：

```
Feb 24 12:05:03 10.0.0.25 snort: [1:1002:2] WEB-IIS cmd.exe access
   [Classification: Web Application Attack] [Priority: 1]: {TCP}
   10.0.0.15:16936 -> 10.0.0.21:80
```

这是一个 Microsoft IIS 攻击。下面的 Perl 脚本片断将解析上述数据：

```
my($a,$b,$c,$type,$class,$priority,$proto,$srcip,$srcport,$dstip,$dst
   port) = $message =~ /\[(.*?):(.*?):(.*?)\] (.*?)\[Classification:
   (.*?)\]\[Priority: (.*?)\]: {(.*?)} (.*?):(.*?) -> (.*?):(.*?)$/;
```

当然，你可以用前一小节的正则表达式验证任何 IP 地址。

9.3.3 Windows Snare

Snare（http://www.intersectalliance.com/projects/BackLogNT/）是用于发现和检测异常系统的一种基于代理的工具。Windows Snare 是用于 Microsoft Windows 的代理，能够读取 Windows 事件日志，并向一台主机转发 syslog 消息。下面是 Windows Snare 事件的一个示例：

```
Oct 29 13:54:50 10.0.1.2 MSWinEventLog 0 Security 238 Fri Oct 29
   13:54:50 2011 593 Security kjs User Success Audit KJS600 Detailed
   Tracking A process has exited: Process ID: 1452 Image File Name: C:\
   windows\system32\ssbezier.scr User Name: kjs Domain: GUARDED Logon
   ID: (0x0,0x2577F) 181
```

根据 Snare 手册，syslog 消息中的每个项目都由一个制表符分隔。下面的 Perl 脚本片断将解析 Windows Snare 消息：

```
my($cruft,$criticality,$sourceName1,$count,$dateTime,$eventId,$sourceN
   ame2,
$user,$sidType,$eventLogType,$computerName,$categoryString,$dataString,
$expandedString,$md5) = split(/\t/,$message);
```

我们使用 split() 函数，因为所有消息都由制表符分隔。split() 返回的变量名相当明显地说明了每个项目的含义。用于设置类型的变量是 $eventLogType，在示例事件中它是“Success Audit”（成功审计）。但是，Snare 文档提供了 Snare 生成的所有可能的事件 ID；上例中的事件 ID（$eventId 变量）是 593。创建一个以 Snare 所知的所有事件 ID 为关键字的 Perl 哈希更有意义。你可以从 http://sourceforge.net/projects/snare/ 获取 Snare。Snare 安装的例子参见第 15 章。

9.3.4 通用 Cisco IOS 消息

Cisco IOS 是用于运行 Cisco 路由器、交换机和其他设备的操作系统。它可以发出数量巨大的各种日志消息。下面是 IOS 安全事件的一个例子：

```
Nov 19 15:32:33 10.4.9.1%SEC-6-IPACCESSLOGP: list 110 denied tcp
   10.0.0.2(113) -> 10.0.0.15(34237), 1 packet
```

下面的 Perl 脚本片段将解析这一事件：

```
my($category,$severity,$type,$srcip,$srcport,$dstip,$dstport) = $x =~
   /%(.*?)-(.*?)-(.*?):.* (.*?) \((.*?)\) -> (.*?) \((.*?)\) /g;
```

几乎所有 IOS 消息都包含一个形式为“% 类别 - 严重性 - 类型”（% category-severity-type）的标识符。上述代码显示严重性为 6。Cisco 定义了如下严重性：

{0| 紧急 } 系统无法使用

{1| 警报 } 需要立即采取行动

{2| 关键 } 关键状况

{3| 错误 } 错误状况

{4| 警告 } 警告状况

{5| 通知 } 正常但是重要的状况

{6| 信息 } 信息性的消息

{7| 调试 } 调试消息。

解析 IOS 消息的更通用方法如下：

```
my($category,$severity,$type,$rest) = $x =~ /%(.*?)-(.*?)-
   (.*?):(.*?)$/g;
```

这将捕获类别、严重性和类型，并将消息的剩余部分放在 $rest 变量中。

关于确定 Cisco 防火墙和 IOS 路由器事故的讨论参见 http://www.cisco.com/web/about/security/intelligence/identify-incidentsvia-syslog.html。

9.3.5 正则表达式性能考虑因素

因为正则表达式基于计算机科学中的非确定性有限自动机（Non-deterministic Finite Automata，NFA）的概念，你必须了解一个事实：正则表达式可能一再影响你所执行的任务的性能。例如，如果你希望从许多正在规范化的日志消息中提取一个子串，你可以编写一个正则表达式提取所需的文本。如果你知道字符串出现在消息的哪个位置，但是不完全确定，这一方法很实用。例如，看看下面的 Perl 脚本：

```
#!/usr/bin/perl
my $text = "I would like to get the following IP address: 10.0.0.2";
for(my $a=0; $a<100000; $a++)
{
   my($IP) = $text =~ m/: (.*)$/;
}
```

我们知道 IP 地址出现在冒号之后。所以我们编写了一个正则表达式，捕获冒号后的所有字符。因为我们在正则表达式两边加上圆括号，Perl 将在数组中返回它所匹配的项（如果有的话）。这就是我们在运行脚本时使用“my($IP) = …”的原因，下面是我们得到的结果：

```
$ time ./regex.pl
real 0m0.727s
user 0m0.724s
sys 0m0.003s
```

执行这个正则表达式 100 万次花费的时间（0.727 秒）看上去不错。运行时间低于 1 秒是很好的成绩。但是，有一种方法可以得到更好的性能，而且完全不涉及任何正则表达式。Perl 有一个 substr() 函数，可以指定偏移位置。从文本字符串中返回子串。现在我们考虑如

下修改过的脚本：

```
#!/usr/bin/perl
my $text = "I would like to get the following IP address: 10.0.0.2";
for(my $a=0; $a<100000; $a++)
{
   my $IP = substr $text, 46;
}
```

因为我们知道 IP 地址在偏移位置 46，所以现在使用 substr() 函数，运行时，看到如下结果：

```
$ time ./substr.pl
real 0m0.103s
user 0m0.100s
sys 0m0.003s
```

运行时间（0.103 秒）比正则表达式好得多。这是因为 substr() 直接到达我们所需的位置。它没有必要搜索字符串进行模式匹配。使用 substr() 只适合于你要提取的数据总是出现在字符串中同样偏移位置的情况。遗憾的是，在实际处理中，事情不总是这么简单。供应商可能使用许多不同的事件格式，IP 地址和端口等内容往往出现在消息的不同位置。这真的是个棘手的问题。

对于倾向于使用 Java 等语言编写自己的解析系统的人们，你可以使用重载的 String 类 indexOf() 方法，寻找一个子串的开始部分。这个方法可以寻找子串的任意部分，不管它在字符串中的什么位置，相当简单。

下一小节进入过程的下一个关键步骤：关联。

9.4 关联

本节的目标是提供丰富的工具和概念使用示例，以便对关联有一个感性认识。提供全面的关联示例超出了本章的范围。创建关联过程的一部分是理解你自己的环境，以及用户、分析人员等的特定需求。

好了，现在我们来定义“关联”一词的含义。《韦氏词典》对关联的定义如下：

> 被关联的状态或者关系；具体地说：现象或者事务之间、数学变量或者统计变量之间存在某种关系，它们倾向于同时变化、相互联系或者同时发生，而这种同时发生的情况不被认为是偶然的。

上述定义有些模糊，不能完全说明日志数据分析中关联的意图。那么，关联对系统和网络专业人士意味着什么呢？它对于日志分析的含义是，我们感兴趣的是将多个相似或者不相似的事件联系起来，形成对更大的事件将要发生的认识，而不是简单地关注单一事件，从而得到所发生的情况的不完整视图。这通过以某种语言创建规则来实现，这些规则建立了安全

和网络管理员和分析人员感兴趣的情况的模型。

例如，如果我们从防火墙和 IDS 接收日志数据，可以用如下规则，捕获在探查尝试之后破坏防火墙策略的行为：

If the system sees an event E1 where E1.eventType=portscan
followed by
an event E2 where E2.srcip=E1.srcip and E2.dstip=E1.dstip and
E2.eventType=fw.reject then
doSomething

上述规则用伪代码表达，详细地说明了两个不相交的事件是如何联系在一起的。该规则查看来自两个不同网络安全系统的两个不同事件。第一个是端口扫描（portscan），由 IDS 检测，下一个事件（称为 E2）是由防火墙生成的。它反映有人企图访问一台主机，这台主机有一个防火墙策略，除了某些受信任的主机之外，排除了其他所有主机的访问。注意，该规则包含了短语“followed by”，这并不意味着该事件必须紧随其后发生，但是有这种可能。上述规则意味着，未来的某个时点必须发生 fw.reject 事件，该规则才能够执行 doSomething 操作（电子邮件、警报等）。应该注意的是，使用 fw.reject 和 portscan 作为 E1 和 E2 的事件类型（EventType）只是对这些事件类型的随意分类（分类学）。

我们已经看到了状态型规则引擎（Stateful rule engine，SRE）的用处。它们为管理员和分析人员提供了创建检测任意顺序或者模式的规则的能力。在这种地方，简单地查看 syslog 文件，自行关联这些事件是不可能的。SRE 的缺点正如它的名称所表现出来的那样，必须维护一个状态的概念，以便执行这些测试。这在上述规则中可以看到，我们比较事件 E1 和事件 E2 的 srcip（源 IP）。除非你有某种方法将事件保存在外设视图或者存储层中，否则无法进行这种关联。

实现的方法之一是简单地使用存档数据存储作为持久化层。SBR 引擎能够简单地挖掘数据库，寻找匹配一条或者多条规则的事件序列。这是代价很高的方案，因为需要进行大量的 SQL 查询并在内存中操作事件数据。你将花费大量的 I/O 操作，影响其他试图使用存档数据库的应用程序。

更好的解决方案是在事件数据进入长期存储之前获取它。在聚合器（如果这也是你进行分析的地方）或者分析服务器接收到数据时，你将数据发送到某种系统消息队列（如 System V 队列），在那里供 SBR 引擎消费和处理，在 SBR 消费事件数据时，它必须将事件保存在内存中，才能够对其应用规则。问题是，如果我们从不让这些事件过期、离开内存，可能耗光系统的 RAM。所以，这就是在数据自身失效之后使事件过期的重要性。

实现事件过期的方法之一是为引擎中的每条规则设置生存时间（TTL）。更确切地讲，我们可以为规则中的每一步设置 TTL。例如，在上述规则中，我们在获得 eventType 为 portscan 的事件时才启动事件跟踪。一旦发生这种情况，就开始计算 TTL。当 TTL 时间已到，如果我们没有看到符合规则下一阶段中需求的事件，可以重置规则，放弃我们为这一规则跟踪的所有事件。

9.4.1　微观关联

微观关联关心的是单一事件或者一组事件中关联的字段，有时候称为原子关联。原始事件数据规范化对于原子关联的高效执行很重要。本节余下部分将提供与原子关联相关的主题和示例。

1. 字段关联

字段关联提供发现规范化事件数据中感兴趣的一个或者多个字段的机制。例如，你可能对 IDS 何时检测到针对 Web 服务器的活动感兴趣。最合乎逻辑的做法是搜索目标端口为 80 或者 443 的事件。

事件类型是另一个字段关联的宝贵来源。许多防火墙使用接受（Accept）、抛弃（Drop）等事件类型。根据事件类型名称关联事件往往很有用。如果你关心对特定服务器的有害访问企图，可以关联事件类型为 Drop，目标 IP 地址为该服务器的事件。

2. 规则关联

规则关联涉及编写一条能够建立某种行为模型的规则。回忆一下前面的伪代码示例：

```
If the system sees an event E1 where E1.eventType=portscan
followed by
an event E2 where E2.srcip=E1.srcip and E2.dstip=E1.dstip and
E2.eventType=fw.reject
then
doSomething
```

这条规则建立一个行为模式的模型。虽然使用的是伪代码，你可以看到这条规则试图寻找一种情况：攻击者对主机进行端口扫描，同时导致防火墙拒绝了他的尝试。

为了使规则关联变得高效，规则引擎至少应该有如下的功能：

- 状态型行为：实施关联的系统有时候称为引擎。而且，规则关联引擎有时候称为状态型规则引擎或者状态型业务规则引擎。这种状态型的操作通过建立安全工程师所感兴趣的行为模型来进行。状态型规则可能由一个或者多个状态组成。例如，一个状态可能寻找目标端口 80（HTTP）的所有事件。如果进入引擎的事件匹配这一准则，规则中的下一个状态可以寻找具有特定源 IP 地址的所有事件（可能用于寻找未授权的 Web 服务器访问）。如果达到第二个状态，我们执行某些行动。
- 计数：计数是一个非常有用也很简单的工具。它实质上使你可以创建一条规则，等待一个事件发生一定次数，然后采取行动。也可以对一个事件的唯一发生进行计数，这也非常有用，因为它可以用于检测某人对一系列不同的目标主机扫描相同的端口。
- 超时：大部分状态型规则引擎实现使用内存进行分析等工作，这有利于速度和效率。我们将引擎使用的内存称为工作内存（working memeory）。随着你在引擎的工作内存

中添加越来越多的数据，不仅使引擎的速度下降，还会影响主机系统的总体性能。对特定事件周期内不匹配某条规则的事件进行定时和老化，对于维护高效的引擎是至关重要的。例如，你可能应该设置 5 分钟的默认老化期。

- 规则重用：很明显，你应该减轻创建规则的负担。从上述的示例规则中可以看到，下面是规则本身的一个条件：If the system sees an event E1 where E1.eventType = portscan。规则可以有许多条件语句。在你所创建的其他规则中重用这类的组件总是很有好处的。
- 优先级：优先级的用处在于可以描绘一条规则或者一组规则执行的顺序。
- 语言：所有规则引擎都实现某种用于指定规则的语言或者伪语言。现在流行的是使用 XML 作为伪语言，但是我们也曾经看到过使用 Lisp 的规则引擎。有些引擎使用专有的格式。要牢记的是，你应该有一种容易学习和使用的语言。
- 行动：毫无疑问，采取行动是规则关联的全部意义。在你的规则引擎中必须有一种执行某种操作的机制，仅仅知道发生了某件事情往往还不够。和规则重用一样，应该有一种机制创建可以由多个规则集重用的行动。

不是所有规则引擎都有上述这些功能。即使你决定要构建自己的引擎，也可以选择其中的一些功能而非全部。意识到成功的规则处理的关键因素很重要。后面的几个小节将讨论构建你自己的规则引擎。

9.4.2 宏观关联

宏观关联关心的是获取其他信息来源，以便进一步验证或者得到事件流上的情报。这种技术有时称为融合关联。宏观关联的例子之一是考虑漏洞扫描数据，并将其与进入分析系统的事件数据进行内联比较。如果你的 IDS 发现一系列网络主机上的端口扫描，知道这些主机是否真正存在、这些主机上是否存在漏洞就很重要了。和微观关联一样，原始事件数据的规范化很重要。这一点也可以扩展到你想要引入关联过程的其他数据集。例如，以用户名做全名映射可能很有用，正如用户角色在特定系统上的作用一样。这种数据称为上下文数据。Windows 或者 Unix 等操作系统将在它们生成的日志消息中插入一些上下文信息。如果你打算获得用户的更详细信息，就需要进行第二个收集步骤，从 LDAP 服务器或者活动目录服务器上获取它们。

本节剩下的部分将先介绍融合关联的概念和例子。

1. 规则关联

和微观关联很相似，宏观关联使用规则关联。主要的不同是微观关联系统中的规则关联往往过渡到一个宏观关联规则中，后者可以用做其他微观规则的输入，或者引发写入文本文件、创建服务台工作单等行动。

2. 漏洞关联

漏洞扫描器帮助寻找容易遭到已知攻击的主机和其他系统。所有扫描器都至少提供如下信息：

- 易遭攻击的主机：主机名称或者 IP 地址。
- 易遭攻击的服务或者端口：Sendmail 端口（25）等。
- 补救步骤：Sendmail 的修补版本等。

大部分扫描器（商业化产品或者其他产品）设计为每天、每周或者每月运行。用户决定扫描是手工运行，还是计划在预设的时间和日期运行。最近 Sourcefire（http://www.sourcefire.com）等 IDS 供应商开始提出实时漏洞扫描的概念。Sourcefire 使用实时网络感知（RNA）作为实时漏洞分析的一种形式。

在威胁分析中，我们可以结合漏洞扫描数据和实时事件数据，以便帮助减少假阳性。例如，如果你的 IDS 检测到网络中的多台主机上发生了端口扫描，它就有可能报告一个被扫描的端口范围。而且，报告的端口中有些可能不活动。如果你对网络进行例行的漏洞扫描，可以使用这些信息和实时获得的 IDS 消息，验证问题中的主机是否真的打开了一个端口，是否容易遭到攻击。

3. 剖面（指纹）关联

从取证的角度看，通过标识抓取（banner snatching）、操作系统指纹、漏洞扫描和远程端口扫描收集信息都是有好处的。而且，这种数据可以用于更深入地了解对你发动攻击的人，对诉讼也有帮助。作为常规关联（例如规则关联）的一部分，可以设置一个行动，当特定性质的事件进入系统时发起这些取证收集任务。然后，收集到的信息可以保存在一个数据库供以后使用。

Nmap 是一种能够执行操作系统指纹识别的工具，可以从 http://www.insecure.org/nmap 取得，下面是 Nmap 运行的一次基本指纹识别：

```
$ sudo nmap -A 192.168.1.6
Starting Nmap 5.21 (http://nmap.org) at 2012-03-25 18:03 EDT
Nmap scan report for 192.168.1.6
Host is up (0.00038s latency).
Not shown: 991 closed ports
PORT STATE SERVICE VERSION
135/tcp open msrpc Microsoft Windows RPC
139/tcp open netbios-ssn
445/tcp open netbios-ssn
2869/tcp open http Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_html-title: Service Unavailable
49152/tcp open msrpc Microsoft Windows RPC
```

```
49153/tcp open msrpc Microsoft Windows RPC
49154/tcp open msrpc Microsoft Windows RPC
49158/tcp open msrpc Microsoft Windows RPC
49160/tcp open msrpc Microsoft Windows RPC
MAC Address: 68:A3:C4:4D:4A:FF (Unknown)
Device type: general purpose
Running: Microsoft Windows Vista|2008|7
OS details: Microsoft Windows Vista SP0 - SP2, Server 2008, or Windows
   7 Ultimate (build 7000)
Network Distance: 1 hop
Service Info: OS: Windows
Host script results:
|_nbstat: NetBIOS name: KEVIN-PC, NetBIOS user: <unknown>, NetBIOS MAC:
   68:a3:c4:4d:4a:ff
| smb-os-discovery:
| OS: Windows 7 Professional 7600 (Windows 7 Professional 6.1)
| Name: WORKGROUP\KEVIN-PC
|_ System time: 2012-03-25 18:04:10 UTC-4
|_smbv2-enabled: Server supports SMBv2 protocol
HOP RTT ADDRESS
1 0.38 ms 192.168.1.6
OS and Service detection performed. Please report any incorrect results
   at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 56.55 seconds
$
```

-A 选项命令 Nmap 执行指纹识别。

4. 反端口关联

大部分防火墙根据允许或者拒绝的数据包和其他事件生成日志消息。反端口关联使用开放端口信息和防火墙数据，以便能够检测“慢速”或者“低”类别的攻击。Nmap 可以用于跟踪系统上打开的端口。下面的伪代码说明基本思路：

```
if (event E1.dstport != (Known_Open_Ports on event E1.dstip))
then
doSomething
```

蠕虫活动是这种技术有助于发现的攻击之一。但是，一般来说，它可以帮助检测攻击者何时尝试访问系统端口或者不存在的服务。有多种跟踪开放端口信息的方法，其中之一是定期运行 Nmap 跟踪开放端口。在你接收事件时，你可以将来自防火墙和 IDS 的目标端口与已知的开放端口作比较。另一种方法是保留一个最新的主机和端口列表，而不是依靠 Nmap。

这种技术只有在你的环境相对静态的情况下可行。

5. 观察列表关联

能够知道何时接收到针对一个主机甚至网络的事件有时候也很有利。反过来也往往一样；你可能应该将攻击的来源放在一个观察列表（也就是攻击者列表）中。

观察列表常常是以从不同位置收集而来的外部情报的形式出现，例如 Dshield（http://www.dshield.org/）有前 10 大攻击者的列表，你可以将它们放在你自己的观察列表中。SANS 也有类似的信息：http://isc.sans.org。

6. 地理位置关联

许多网络管理平台使用网络地图来直观地展示企业中的设备。生成警告时，地图上的节点发亮，以表示出现了问题。这类地图在安全界没有多少价值，但是在威胁发生时实时地用图形表示的能力确实有价值。这需要一些不同寻常的处理。美国互联网号码注册机构（American Registry for Internet Numbers，ARIN）是对这一处理有益的资源。ARIN（http://www.arin.net）是北美、加勒比海地区和非洲赤道附近地区的互联网号码注册机构。互联网提供商、公司等通过 ARIN 进入网络。ARIN 收集这些组织的联络信息，虽然它不收集经纬度信息，但是你可以获取机场代码或者从 WHOIS 记录中反向查找 IP 地址。ARIN 网站有许多工具，你可以使用这些工具查询它的服务，找到关于 IP 地址和网络的信息，特别是联络信息。虽然根据这个信息在地图上绘制出攻击点不是 100% 准确的，但是对于追踪作恶者大有帮助。VisualRoute 是一个可用来直观地描绘数据包从一个位置到达另一个位置的路径的工具，相关信息参见 http://www.visualware.com/personal/products/visualroute/index.html。

9.4.3　使用环境中的数据

关于环境的数据是关联数据的最佳来源之一。正如我们在 9.4.2 节宏观关联部分中所提到的，这些数据可以视为上下文。例如，如果你知道公司的假日安排，可以使用这一信息，在每个人都不上班的时候发现内部资源的访问时，发出一个警报。你可以使用的较为常见的环境触发因素包括：

- 休假时间表
- 业务时间
- 假日计划
- 内部资源访问权限
- 重复的网络“事件”例如漏洞扫描
- 计划的系统、数据存储备份等
- 维护安排，例如路由器配置更改和重启、操作系统补丁等

上述因素可以用于帮助确定某些行为是合法，还是超出了正常的界限。

9.4.4 简单事件关联器

简单事件关联器（SEC）是一种基于 Perl 的工具，实现一个简单的关联系统，它可以在 http://kodu.neti.ee/~risto/sec/ 取得。

SEC 开发用来填补开放源码日志分析工具和商业化系统之间的空白。它具备创建规则的健全语法，能够完成从简单到非常复杂的工作。规则可以链接起来组成行为模式，这可以用于检测已知的情况，帮助诊断或者发现未知的情况。SEC 引擎的核心是上下文的概念。上下文可以看作一个事件或者一系列事件发生的占位符，它的概念使得 SEC 可以状态型的风格运作。这对于检测多种日志消息模式是非常关键的，这些模式组合起来能够表现出令人感兴趣的一些情况，我们将很快看到一些实用示例。

1. SEC 基础知识

本节的目的不是完整地介绍 SEC，而是介绍关键的特性。SEC 的安装非常简单，所以我不说明这一过程，而是介绍实用的 SEC 规则示例。现在，我们先来看看 SEC 支持的各种规则类型：

- Single——匹配一个事件并执行一项行动。
- SingleWithScript——匹配一个事件并从它所运行的一个脚本或者程序中获得返回码。
- SingleWithSuppress——匹配一个事件，然后忽略后续匹配 t 秒。
- Pair——将给定事件范围内的两个或者更多事件组合为单一事件。
- PairWithWindow——类似于 Pair，但是为下一事件到达等待 t 秒。
- SingleWithThreshold——计算一个时间窗口中匹配的事件数量，直到到达某个阈值。一旦达到阈值则触发一个行动，在剩余时间窗口不考虑其他任何事件。
- SingleWith2Thresholds——计算时间窗口 t1 内的匹配事件数量，并在达到第一个阈值时执行一个行动。计数被重置并在 t2 秒钟内计算匹配事件数量，如果在时间窗口 t2 中计数低于第二个阈值，执行另一个行动。
- Suppress——不匹配事件，也不执行行动。
- Calendar——在指定时间执行行动，配置类似于 UNIX 的 crontab 条目。

SEC 规则基本格式如下：

```
type= <Rule Type>
ptype=RegExp
pattern= <regular expression pattern>
desc= <description text>
action= <semicolon separate action list>
```

我们将会看到一些实际的例子。SEC 支持许多行动，但是这里只介绍较为通用的：

write< 文件名 >[< 事件文本 >]：write 命令将把事件文本写入文件名所指的文件。使用连字符（“-”）可以写入到标准输出。

shellcmd<shell 命令 >：执行一个 shell 命令或者程序。

spawn<shell 命令 >：这和 shellcmd 完全相同，但是 shell 命令或者程序的输出会反馈到 SEC 中。这意味着，你可以编写 SEC 规则关联这个输出。

event[< 时间 >][< 事件文本 >]：这导致创建 SEC 内部事件，可以用作 SEC 的反馈。其中的时间是 SEC 等待事件文本反馈的时间数。注意，如果你在时间上使用非零值，SEC 将等待到这个时间后才做其他的工作。在大部分情况下，可以使用零值，SEC 将继续其处理。

2. 实际的例子

现在我们想要做的是用几个实际的例子介绍 SEC 的用法。这些例子都很容易扩展，用于你自己的环境。

9.4.5 状态型规则示例

回忆一下本章前面的伪代码规则：

If the system sees an event E1 where E1.eventType=portscan
followed by
an event E2 where E2.srcip=E1.srcip and E2.dstip=E1.dstip and
E2.eventType=fw.reject
then doSomething

我们感兴趣的是捕捉两个单独的事件。一个是 IDS 端口扫描事件，另一个是防火墙拒绝事件。如果两个事件的源和目标 IP 地址相同，我们就要进行某些操作。为此设置的 SEC 规则如下：

```
#
# Rule 1: Catch a portscan from source IP address to destination IP
# address. It will then create a dynamic context based on whatever the
# source and destination IP addresses are.
#
type=Single
ptype=RegExp
pattern=portscan (\d+\.\d+\.\d+\.\d+) -> (\d+\.\d+\.\d+\.\d+)
desc=$0
action= create PORTSCAN_$1_$2;
#
# Rule 2: Catch a firewall reject from source IP address to destination
   IP
# address. If the portscan context already exists, we have a situation
# where an attacker is portscanning AND getting a reject from a
   firewall, all
# from the same source IP address to the same destination IP address.
```

```
#
type=Single
ptype=RegExp
pattern=reject (\d+\.\d+\.\d+\.\d+) -> (\d+\.\d+\.\d+\.\d+)
context=PORTSCAN_$1_$2
desc=$0
action= write - Portscan and firewall reject from $1 to $2;
```

例子中有两条规则。乍看，你可能疑惑于我们为什么不使用 Pair 规则。Pair 规则的问题是它只检测一个事件紧接着另一个事件发生的情况。我们希望寻找一个事件的第一次发生，然后寻找过后一段时间发生的第二个事件，而不一定是紧接着发生的事件。

无论如何，规则中的注释已经很好地描述了所发生的情况。第一条规则捕捉一个端口扫描事件，并且获得源和目标 IP 地址。然后，它创建一个动态上下文，形式为 PORTSCAN_SRCIP_DSTIP，当然其中的 SRCIP 和 DSTIP 就是实际的 IP 地址。它还使用了 SEC 特殊变量 $0，这是匹配模式的整条日志消息。第二条规则搜索拒绝事件，提取源和目标 IP 地址，并用它们了解是否存在相同源和目标 IP 地址的端口扫描上下文。如果发生这种情况，向标准输出写入一条消息。我们来简略地看看 SEC 运行的一个示例。测试规则最好的方式是运行 SEC，让它接受标准输入的内容作为输入这用命令行参数 -input=- 完成。你用“–conf= 文件名”选项指定规则文件。在下面的例子中，我们对 SEC 的输入用粗体写出：

```
$ ./sec.pl -conf=portscan-reject.rule -input=-
Simple Event Correlator version 2.2.5
Reading configuration from portscan-reject.rule
2 rules loaded from portscan-reject.rule
portscan 10.0.0.2 -> 10.0.0.3
portscan 10.0.0.2 -> 10.0.0.4
portscan 10.0.0.2 -> 10.0.0.5
portscan 10.0.0.2 -> 10.0.0.6
Some random text
More text..
reject 10.0.0.2 -> 10.0.0.4
reject 10.0.0.2 -> 10.0.0.6
reject 10.0.0.2 -> 10.0.0.5
reject 10.0.0.2 -> 10.0.0.3
Almost done..
There..
Creating context 'PORTSCAN_10.0.0.2_10.0.0.3'
Creating context 'PORTSCAN_10.0.0.2_10.0.0.4'
Creating context 'PORTSCAN_10.0.0.2_10.0.0.5'
Creating context 'PORTSCAN_10.0.0.2_10.0.0.6'
```

```
Writing event 'Portscan and firewall reject from 10.0.0.2 to 10.0.0.4'
   to file -
Portscan and firewall reject from 10.0.0.2 to 10.0.0.4
Writing event 'Portscan and firewall reject from 10.0.0.2 to 10.0.0.6'
   to file -
Portscan and firewall reject from 10.0.0.2 to 10.0.0.6
Writing event 'Portscan and firewall reject from 10.0.0.2 to 10.0.0.5'
   to file -
Portscan and firewall reject from 10.0.0.2 to 10.0.0.5
Writing event 'Portscan and firewall reject from 10.0.0.2 to 10.0.0.3'
   to file -
Portscan and firewall reject from 10.0.0.2 to 10.0.0.3
```

我们简单地将输入数据放到一个文件中，并且从该文件剪贴到 SEC 的标准输入。在我们输入之后，看到 SEC 创建了 4 个单独的上下文，对应 4 个端口扫描事件。在每个拒绝事件进入 SEC 时，它开始为每个不同的端口扫描和拒绝事件配对发出消息。这是很有价值的区分，SEC 可以检测出这种情况，而不管匹配的端口扫描和拒绝事件之间发生了什么类型的事件。应该指出，端口扫描和拒绝事件的格式是人为设计的，只为了简化这个例子。在实际工作中，你需要编写更加复杂的正则表达式，以捕获这些信息。

1. 漏洞关联示例

假定你想要执行漏洞关联，减少假阳性。这意味着，你使用 Nessus 之类的工具定期扫描你网络，寻找有漏洞的系统。扫描器输出被放入一个数据库，我们可以在 SEC 中编写一条检测端口扫描的规则，运行一个程序查询漏洞数据库，并返回受攻击系统是否有漏洞。下面是实现上述功能的 SEC 规则：

```
#
# Rule 1: Check to see if the attacked system is vulnerable
#
type=SingleWithScript
ptype=RegExp
pattern=portscan \d+\.\d+\.\d+\.\d+ -> (\d+\.\d+\.\d+\.\d+)
script=./vulnCheck.pl $1
desc=$0
action=write - $1 is vulnerable!
action2=write - $1 is NOT vulnerable
```

SingleWithScript 规则执行一个脚本或者程序。如果程序的退出码为 0，则采取行动。如果退出码非 0，则执行操作 2。Perl 脚本 vulnCheck.pl 简单地执行 exit(0)，这将导致行动被执行，换言之，我们拥有一个有漏洞的系统。

下面是运行的一个示例：

```
$ ./sec.pl -conf=vulnerability.rule -input=-
Simple Event Correlator version 2.2.5
Reading configuration from vulnerability.rule
1 rules loaded from vulnerability.rule
portscan 10.0.0.2 -> 10.0.3.4
Child 25163 created for command './vulnCheck.pl 10.0.3.4'
Child 25163 terminated with exitcode 0
Writing event '10.0.3.4 is vulnerable!' to file -
10.0.3.4 is vulnerable!
```

2. Cisco IOS 配置变更

许多环境部署了某种配置变更控制。例如，所有硬件和软件的变更在某几个小时之间进行。这种控制可以给你带来好处，如果你在正常时间之外接收到路由器配置的变更，就需要担心所发生的事情了。SEC 可以用于检测这种情况，我们来看看捕获这种现象的配置示例：

```
type=Single
ptype=RegExp
pattern=%SYS-\d+-CONFIG
desc=$0
action= write - A configuration change has occurred!
```

初看，这是完全没有问题的处理方法。但是我们真的想要在配置期间（甚至在你的维护期间）都触发行动吗？如果你对此不在乎，可以原封不动地使用上述 SEC 配置。但是对于想要限制警报的人来说，可能会对如下的配置感兴趣：

```
type=SingleWithScript
ptype=RegExp
pattern=%SYS-\d+-CONFIG
script=./isItMaintTime.pl
desc=$0
action=write - Configuration during non-maintenance period!
action2=write - Configuration attempt during maintenance period!
```

这个 SEC 配置块将运行一个 Perl 脚本，检查接收事件的日期和时间是否和维护时间一致。我们还可以更进一步：

```
type=SingleWithScript
ptype=RegExp
pattern=^.* (\d+\.\d+\.\d+\d+) .*%SYS-\d+-CONFIG: .* from (.*?)$
script=./isItMaintTime.pl $1
desc=$0
action=write - $1 Configured during non-maintenance period from $2!
action2=write - $1 Configured during maintenance period from $2!
```

现在，我们提取想要配置的主机，然后将它传递给 isItMaintTime.pl 脚本。目标是查找（可能在数据库中）问题路由器在当前维护期内是否安排了进行配置。大部分 syslog 消息包含了生成它的主机。第一个正则表达式试图收集这一信息。注意，你可能需要关闭 DNS 解析，使路由器的 IP 地址不被转换为主机名。最后，收集配置的来源，就可以把配置来源加入报告中。

3. 混杂模式检测

当网络接口被置于混杂模式时，所有数据包都被发送到内核处理，包括目标不是该网络接口卡 Mac 地址的数据包。这样做很糟糕，主要原因之一是，在使用混杂模式网卡的系统上的用户可以使用嗅探器等工具，查看所有的网络数据包。我们来看看在 Linux 上如何检测进入混杂模式的接口：

```
type=Single
ptype=RegExp
pattern=^.*(\d+\.\d+\.\d+\.\d+).* (.*?) entered promiscuous mode$
desc=$0
action= write - Interface $2 on $1 entered promiscuous mode
```

上述 SEC 规则检测主机和主机上进入混杂模式的网卡。

4. 需要寻找的关键字

下面的 SEC 示例展示了一些值得注意的关键字。

内核的混乱通常意味着某种硬件故障或者类似的事件。SEC 可以用如下规则检测这些事件。

```
type=Single
ptype=RegExp
pattern=panic
desc=$0
action= write - A potential kernel panic has occurred
```

应该在这些事件中捕获致命的消息，下面的 SEC 规则完成这种捕获：

```
type=Single
ptype=RegExp
pattern=fatal
desc=$0
action= write - A fatal error may have occurred
```

最后，要留意其中包含字符串“password”或者“passwd”的消息，因为它们可能表示破坏密码维护正常渠道的恶意行为。下面的规则检测密码消息：

```
type=Single
ptype=RegExp
pattern=password|passwd
```

```
desc=$0
action= write - Password operation detected
```

5. 应用程序退出码

应用程序退出时，通常向操作系统返回一个错误码。这个代码一般为 0，表示正常程序结束。退出状态 1 通常用于表示发生了某种异常行动，使得应用程序突然中断。下面的 SEC 规则将检测这些错误码：

```
type=Single
ptype=RegExp
pattern=exit status 1
desc=$0
action= write - An application exited with status of 1
```

当然，我们可以将上述规则扩展为：

```
type=Single
ptype=RegExp
pattern=exit status (\d+)
desc=$0
action= write - An application exited with a status of $1
```

这些规则的好处是真正获取退出码，我们可以对其进行更细粒度的分析。

6. SEC 小结

最后，我们还需要指出一点。在本章的大部分篇幅中，我总是说到需要规范化原始数据以便进行关联。利用 SEC 则不是这样的。SEC 最好的工作方式是通过读取日志文件。但是，你可以通过其他的方法将数据传递给 SEC。所以，你可以用它关联你的日志数据，或者在收集并规范化数据之后用它关联数据。

9.4.6 构建自己的规则引擎

如果你正在使用没有规则引擎的日志分析系统，该怎么办？如果你对编写自己的引擎感兴趣，且倾向于 Java，本节介绍两个框架，你可以用它编写两种不同的处理引擎：

- 基于规则
- 基于流

1. 使用 Jess 的基于规则的引擎

Jess 由 Sandia National Labs 维护（http://herzberg.ca.sandia.gov/Jess/），它不是开源的，但是可以免费用于非商业和学术用途。对于大部分组织来说，它的商业许可证模式也不是很昂贵。

了解 Jess 的最佳资源之一是《Jess In Action》一书，该书由 Jess 的创造者和维护者 Ernest Friedman-Hill 博士编著。本节只介绍 Jess 用于安全日志分析的功能，Jess 常规使用方法的详细解释应该参见该书。

Jess 用 Java 编写。使用它并不真的需要了解 Java，因为 Jess 有自己的规则语言，并且可以像 Shell 那样运作（名称就来源于此）。但是，为了以日志数据收集和规范化相一致的风格使用它，你必须进行一些 Java 开发。所以，本节假定你对 Java 开发有一些了解。你不一定要成为一位大师，但是应该熟悉 Java。

（1）Jess 和 Rete 算法

作为规则引擎，Jess 能力的核心是实现 Rete 算法（注意，Rete 的发音是“ree-tee”）。Rete 算法是 Charles L. Forgy 博士发明的，该算法的出现是为了解决模式匹配问题。用于规则上下文或者规则引擎时，传统的模式匹配相当低效。据 Friedman-Hill 博士所说，Rete 算法通过记忆已经断言或回收到工作内存中的数据实现其目标（很快将会有更详细的说明）。这意味着，规则引擎只根据实际改变的数据项来测试（或者触发）规则。这减少了引擎所必须执行的工作量。规则通常有所谓的左手边（LHS）和右手边（RHS）。LHS 是条件，类似于你所习惯的 if-then-else 语句。RHS 是 LHS 为真时采取的行动，这很简单。

Rete 算法将其数据组织为类似树形的结构。树中的每个节点代表利用添加或者从内存删除的数据进行的一次测试。每个节点有一个或者两个输入，可能有多个输出。当新的数据进入树中，它通过树进行过滤，直到到达一个终端节点。到达终端节点意味着用所有导致规则为真的数据点创建了一个新的活动记录。规则越详尽，Rete 系统工作得越好，也就是说，你应该编写规则在给定数据中搜索非常详细的项目。这在 Rete 树中形成某种确定方式的一条遍历路径，展现了 Rete 算法固有的高效率。

基于 Rete 的规则引擎很多。谈到这些引擎时常常用到某种术语学，所以我们现在要讨论它们。

演绎和归纳。演绎（forward-chaining）系统中规则依靠一组数据执行。换句话说，它是数据驱动的。归纳（backward chaining）系统查询一个数据集获得真实值。这个数据集几乎总是预先存在的，所以归纳法不一定是数据驱动的。Prolog 是归纳法的经典例子，Jess 可以这两种模式运作。

工作内存（working memory，WM）是规则引擎中的一个区域，保存由一条或者多条规则处理的数据。该区域有时候称为事实空间。维护高性能规则引擎的关键是在 WM 中保存一些事实。你无法不断地在 WM 中放入事实而不清除它们。这不仅会导致 Java 进程消耗内存和进行磁盘交换，而且 Rete 算法的性能也将开始下降。

对象断言（object assertion）。数据通过被称为“断言”（Assertion）的过程放入 WM。我们使用对象这一术语是因为本节提供的例子将 Java 对象放入 WM。这可以编程或者由规则语言自身完成。

对象回收（object retraction）。WM 中的数据通过称为“回收”的过程删除。这可以编程

或者由规则语言自身完成。

议程（agenda）是具有将被执行 / 触发的 RHS 的规则的列表。

冲突解决（conflict resolution）是选择议程中的行动、加以触发的过程。程序员不能完全直接控制这个过程，设置突出（Salience）值是影响这个过程的方法之一。突出是优先级的一种形式。你可能拥有条件类似的规则。规则引擎必须决定 WM 中的同一个事实满足两条规则时触发哪一条。为某条规则设置较高的突出值（通常是有符号整数）将影响引擎执行这条规则，而不是另一条候选规则。

（2）Jess 规则语言

Jess 使用类似于 Lisp 的语言来编写规则。你必须使用它来归纳你的想法，但是只要你动手去做，就会发现这种语言富有表现力且功能强大。我们将简单地介绍这种语言。《Jess In Action》一书很好地描述了它。在 Jess 网站上也有一个免费的手册，描述了该语言的语法细节。

JavaBeans 惯例

取值方法和设值方法

注意，取值方法（getType()、getDstPort() 等）的命名惯例实际上是常见的 JavaBeans 属性命名惯例。Jess 使用对应于取值方法名称的属性名称，有时候用“is”代替“get”，例如用于布尔属性时。

在此，我们只打算介绍最基本的语法细节，说明如何实现安全日志数据分析目标：

```
(defrule find-port-80
?event1 <- (event (dstport 80))
=>
(printout t "Found an event with dst port of 80: " (fact-slot-value
   ?event1 type) crlf)
(retract ?event1))
```

我将描述规则的每个部分。首先，defrule 标记为规则命名。?event1 <- (event (dstport 80)) 是规则的左手边（LHS）。它取得名为 event 的对象，调用取值方法 getDstport()，检查该属性是否为 80，并将事件对象赋予一个变量 ?event1。注意，我们去掉了取值方法中的“get”。在 Jess 中，你只需要使用取值方法中 get 之后的部分。所以，这意味着在你的 Java 代码中，所有取值方法都需要加上 get 前缀。

=> 符号定义规则的 RHS。对于在学校中学习过逻辑或者离散数学课程的人们来说，=> 操作符可以看作一个结论，或者暗示着一个证据陈述或者参数。如果 LHS 的值为真，则执行该结论之后的所有操作。在上述例子中，用 printout 操作符打印一些文本。在 printout 命令中，使用 fact-slot-value 操作符访问事件对象的一个成员。和 dstport 一样，type 是时间对象中 getType() 对应的属性名。最后，我们调用 ?event1 上的回收操作，从 WM 中删除对象。

(3) Jess 的实际应用

为了使用 Jess 分析日志数据，我们必须进行一些 Java 编程。首先，我们假定日志聚合和规范化已经完成。下面的 Java 类 SecurityEvent 是代表单一日志事件的骨干对象：

```
public class SecurityEvent
{
   private String mSrcip;
   private String mDstip;
   private int mSrcport = 0;
   private int mDstport = 0;
   private String mType;
   private String mHostname;
   /*
   Constructor
   */
   public SecurityEvent(String hostname,
         String dstip,
         String srcip,
         int dstport,
         int srcport,
         String type)
   {
   mSrcip = srcip;
   mDstip = dstip;
   mHostname = hostname;
   mDstport = dstport;
   mSrcport = srcport;
   mType = type;
   }
   /* member data getters */
      public String getType()
   {
      return mType;
   }
      public String getHostname()
   {
      return mHostname;
   }
      public String getSrcip()
   {
```

```
        return mSrcip;
    }
        public String getDstip()
    {
        return mDstip;
    }
        public int getDstport()
    {
        return mDstport;
    }
        public int getSrcport()
    {
        return mSrcport;
    }
}
```

该类有普通的私有成员和这些成员的取值方法。我们忽略了设值方法，因为你通常不需要修改日志事件细节。但是，如果你需要这种功能，可以自行添加。

代码的下一部分说明如何使用 Jess 和事件对象：

```
import Jess.*;
public class Main
{
    public static void main(String[] argv)
    {
        try
        {
            //Create new Rete instance
                Rete rete = new Rete();
            //Read in the rules file from file
            rete.executeCommand("(batch myrules.clp)");
            //Define our Java SecurityClass as a Jess class called event
            rete.defclass("event," "SecurityEvent," null);
            //Create an new instance of a SecurityEvent and assert it
    rete.definstance("event,"
                new SecurityEvent("webserver," //hostname
                        "10.0.0.2," //dst ip
                        "10.0.0.3," //src ip
                        80, //dst port
                        5512, //src port
                        "Web-Access-Event" // event type
                        ), false);
```

```
            // Run the engine
            rete.run();
        }
        catch(JessException jex)
        {
        jex.printStackTrace();
        }
    }
}
```

嵌入的注释很容易理解。本质上，我们从一个磁盘文件中加载规则文件；这通过 executeCommand() 方法完成，它试图加载一个名为 myrules.clp 的文件。defclass() 方法告诉 Jess 我们的事件类的相关情况。该方法的第一个参数定义了如何应用规则文件中的对象 definstance() 方法断言一个进入 WM 的新 SecurityEvent 对象。最后，run() 方法运行引擎。在生产规则引擎中，你应该在循环中断言对象和运行引擎，可能从某种队列或者数据库中获取事件对象。根据你计划每秒断言的事件数量，可能需要在每次对象断言时运行引擎，也可以先断言一定数量的事件，然后运行引擎。

另一种替代方法是在一个线程中断言事件，然后用 Rete.runUntilHalt() 在专用线程内持续运行引擎。这完全取决于你的事件发生速度、拥有的规则数量、部分匹配的规则数量（我们将对部分匹配做更多的介绍）等。你可能需要尝试自己的引擎，了解最适合你和你的环境的方案。回忆前一小节中的规则，我们假定你要寻找一种事件之后发生的另一种事件，也就是状态型规则的例子：

```
(defrule find-portscan-followed-by-reject
?event1 <- (event (type portscan) (srcip ?sip))
?event2 <- (event (type reject) (srcip ?sip))
=>
(printout t "Portscan followed by Firewall Reject from: ?sip" crlf)
(retract ?event1 ?event2))
```

上述规则有两个状态，一个搜索类型为 portscan 的事件。第二个状态搜索 reject 事件。第一个状态获取事件的源 IP 并将其赋予 ?sip；第二个状态使用 ?sip，与第二个事件的源 IP 比较，如果匹配（也就是同一个人导致了 portscan 和 reject 事件的发生）则触发这条规则。LHS 的最后一个行动是从 WM 回收 ?event1 和 ?event2。

我们已经提到了部分匹配的一些相关知识。对于任何基于 Rete 的引擎都需要了解这个问题。我们假设在上述的规则里，有 5000 个匹配第一种状态的事件通过引擎。这意味着引擎必须在 WM 中保存这 5000 个部分事件。当一个匹配 RHS 中第二个条件的事件进入引擎时，规则完成并触发。

这个例子并不完美。上述规则只匹配随后立刻发生 reject 事件的 portscan 事件。在某种情况下，这可能就是你想要的结果，但是你肯定希望有比这更灵活的方法。reject 事件可能在 portscan 事件发生之后几分钟才发生。为了补救这种情况，你需要在 SecurityEvent 对象中添加一个时间戳，这个时间戳应该采用 long 类型。你可以使用 System.currentTimeMillis() Java 调

用来设置这个值。你还需要添加一个对应的取值方法。一旦这么做，规则就变成这样：

```
(defrule find-portscan-followed-by-reject
?event1 <- (event (type portscan) (srcip ?sip) (timestamp ?time1))
?event2 <- (event (type reject) (srcip ?sip) (timestamp ?time2))
(test (>= ?time2 ?time1))
=>
(printout t "Portscan followed by Firewall Reject from: ?sip" crlf)
(retract ?event1 ?event2))
```

规则中的第一个状态获取对象的时间戳，并将其赋予 ?time1。第二个状态将事件 2 的时间戳赋予 ?time2。新规则引入 test 函数，该函数使用 >= 运算符确定 ?event2 发生在 ?event1 的同时或者以后。

（4）Jess 的性能

据 Friedman-Hill 所说，确定 Rate 引擎的性能特性很困难。他写到，引擎的运行时间大约与如下值成正比：$R'F'^{P'}$。R' 是一个小于规则数量的数字。F' 是引擎每次迭代中改变的事实数量。P' 是大于 1、小于规则平均模式数量的一个数。

（5）Jess 小结

本节只介绍关于 Jess 的最基本细节。重点是使用 Jess 进行安全日志分析。Jess 的系统占用非常小，很容易嵌入任何 Java 应用程序（包括 J2EE）中。

2. 使用 Esper 的基于流的引擎

最后一个开发者为中心的示例采用基于流的处理，或称复杂事件处理（Complex Event Processing，CEP）。基于流处理的思路是，数据流（或者事件）通过一个 CEP 引擎，可以从多个事件中发现复杂的模式。CEP 引擎采用查询语言，允许你定义感兴趣的模式。理解 CEP 引擎最好和最简单的手段是将其与一个数据库比较。数据库使用查询筛选数据。CEP 引擎让数据流过查询，寻找有趣的现象。CEP 引擎通常非常快速，基准测试的值是每秒可有数十万个事件通过数千条规则（查询）的评估。这是如何做到的？ CEP 引擎执行的大部分工作在内存中而非磁盘中进行。这样做的好处是速度快，但是也意味着，如果你预测到有许多事件和查询，就需要许多 RAM 和 CPU。本节将介绍 CEP 的基础知识。网上有许多好的资源，更深入地介绍它。

我们来看看图 9.2，它描绘了通过 CEP 引擎的基本流程。

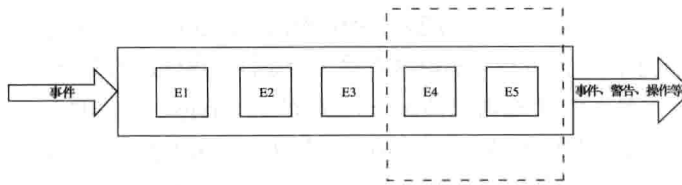


图 9.2　CEP 引擎基本流程

在图 9.2 中我们可以看到，事件流入 CEP 引擎。输出的是事件本身以及根据发现的模式创建的警报或者行动。标签为“Window”的方框指出了流处理的暂时性。利用开窗（Windowing），你可以定义搜索事件模式所需要的事件长度限制，在此窗口之后，引擎将清除保存在内存的任何事件。

Esper（http://esper.codehaus.org/）是最有名的开源 CEP 引擎，它是一个框架，允许你快速地用少量代码构建一个 CEP 引擎。简单地说，Esper 有如下特征：

- 纯 Java
- 可以嵌入到任何 Java 应用程序。
- 通过 Esper 查询语言（EQL）表达，这是一种类似 SQL 的定义语言。
- 有很好的文档

除此之外，还可以获取 Esper 的商业支持。

现在，你可能会问 CEP 对关联有何帮助。让我们来看一个简单的例子。和 Jess 一样，你需要创建 Java 类，建立你计划要分析的规范化日志消息种类的模型。在下面的例子中，我们使用两个单独的对象，但是还有其他的选项。

好了，我们来看看下面的 EQL 语句：

```
select * from pattern [every LoginFailure -> (timer:interval(10 sec)
   having count(*) >= 5 and LoginSuccess)]
```

上述 EQL 语句搜索一个特定的模式。首先，它搜索 LoginFailures（登录失败）。我们将窗口设置为 10 秒。然后，我们在这个窗口中搜索至少 5 个 LoginFailures。10 秒钟的 5 次失败之后必须接着有一个 Login Success（登录成功）。我们所要搜索的可以描述为：

如果 10 秒钟发生 5 次登录失败，然后有一次登录成功，则发出警报

这是暴力破解登录尝试的一种经典模式。当然，在现实的应用程序中，我们需要通过来源和用户，确保 LoginFailures 和 LoginSuccess 相互对应。

流处理是基于规则引擎的一个确实可行的替代品。我们只是简单地介绍了使用 CEP 进行关联的方法。Esper 有许多出色的文档，可以在 http://esper.codehaus.org/tutorials/solution_patterns/solution_patterns.html 找到大量的解决方案模式。

9.5　常见搜索模式

有一些常见模式能够帮助发现独特的关联。表 9.1 介绍了较为常用的一些模式。

表 9.1　常见模式

模　式	讨　论
X 次登录失败后有一次登录成功	这可能表示暴力破解的登录尝试。X 应该取什么值？我们已经发现 5 和 10 之类的数字得到采用。在 Windows 上，当某人登录，Windows 将会生成不只一条登录消息。你应该登录到自己的工作站，看看生成的登录事件数量，在你的环境中调查这一情况。一旦确定所有额外的消息，应该使用日志分析工具过滤其他的登录消息

（续）

模　式	讨　论
创建非管理员账户之后进行权限提升	这不是典型的用户管理路径，也就是说，非管理员账户一般不会将其权限提升到管理员级别，或者其他高级用户级别
VPN 用户在工作时间外 / 内登录，并向网络之外传输数兆字节或者更多（在 VPN 连接期间）数据	这可能表示数据外流，实现起来较为困难，因为需要使用 Netflow 等工具。实时收集、总结和关联 Netflow 并不容易，可能需要以事后处理的方式进行
网络上的一台主机开始攻击或者探查网络上的其他主机	这可能表示源主机受到蠕虫 / 木马 / 恶意软件等的感染。你可以使用防火墙日志确定
在很接近的时间内 X 次尝试访问用户没有权限的共享 / 文件 / 目录等	这可能表示用户账户入侵。如果发生在下班时间，应该立即调查
从同一个工作站以多个用户名登录	这相当简单。某人正在用多个不同的用户名访问网络上的多个不同资源
在多个系统上有多个防病毒软件失效	这种模式较为普遍，但是如果你开始在系统或者主机上看到广泛的防病毒软件失效，可能表示病毒爆发
攻击 DMZ 系统，随后有出站连接（经典！）	这可能代表着僵尸感染
攻击 DMZ 系统，随后在同一个系统上更改配置	攻击者获得系统访问权限并进行修改
在几分钟内有许多 Web 404、401、500 和其他 Web 错误码	这可能表示你的 Web 服务器或者应用程序因为各种原因无法正常运作

9.6　未来

未来会怎样？实际上，未来相当光明。大数据运动带来了存储和处理大量数据的新方法。在大数据的世界里，你能够在多台具有板载磁盘存储的商用机器上存储大量数据。存储数据中的筛选通过运行作业来实现，这些作业用于寻找你所感兴趣的内容，然后返回与你的需求紧密相关的结果集。这种系统合乎逻辑的用途是搜索你已知或者未知的模式（听起来很熟悉吧？）。

9.7　小结

本章我们介绍了许多内容，目的是为你提供实用的工具和围绕过滤、规范化和关联的概念。网络和安全管理员会发现本章介绍的主题很令人兴奋。日志数据处理的一大部分（至少在有目的的处理中）是以能够帮助你理解数据含义的方式对数据进行处理。

参考文献

Ranum, M. J. (1997). Artificial ignorance: how-to guide, September 23. Web, March 19, 2012. <http://www.ranum.com/security/computer_security/papers/ai/>.

第 10 章 Chapter 10

统计分析

本章要点

- 频率
- 基线
- 机器学习
- 结合统计分析和基于规则的关联

10.1 概述

第 9 章中讨论的主题之一是基于规则的关联。这类关联非常有用，遗憾的是，它基于创建规则，捕捉你已经略知一二的事物。为了捕捉你尚不知晓的事物，或者捕捉看上去与其他事物类似的事物，你需要采用一些统计方法。本章和第 11 章接触统计概念较常用的一些使用方法，帮助你根据环境中的日志数据收集发现正在发生的一些事件。在本章中，我们关注频率、基线、机器学习以及统计分析与基于规则的关联相结合的方法。

10.2 频率

可以使用一种非常简单的技术在所有日志来源中计算输入的日志流中的源和目标 IP 地址。思路相当简单：一个目标 IP 地址发生事件数量的高峰可能是某种情况（例如某人针对该系统发动攻击）发生的早期预警标志，这对源 IP 也成立。这种思路很有效，原因是许多安全

设备和系统记录源和目标 IP 地址。即使你没有获得源 IP 信息，也知道潜在攻击的目标（你应该知道，这是由防火墙、IDS 或者其他检测 / 控制系统监控的设备），你往往需要对其展开进一步的调查。

在这种形式的分析中，时间是一个因素。你不应该简单地持续对目标 IP 地址和发生次数进行计数。在某个预定时间周期之后（越短越好），应该考虑清空目标或者源 IP 累积的频率数值，重新开始累加。也就是说，特定目标 IP 地址的事件发生数量应该完全重新累积。这使得发生频率较低的主机不会出现在你的“雷达”中，而在同样时间跨度中频率较高的主机以更为明显的方式出现在雷达上。

虽然频率是简单的分析技术，但是它可能不符合要求。我们需要的是在发生“正常”之外的某种情况下，更为自动化的发现技术。这就是基线起作用的地方，我们将在下一小节讨论它。

10.3 基线

顾名思义，基线为你提供一组数据的标准度量。为什么我们需要基线？这有多种令人信服的原因，包括：

- 增进态势感知（SA）
- 发现新威胁
- 在你已经部署的设施上获得更好的投资回报（ROI）
- 从你的数据获得额外的含义，以便自动采取措施
- 通过指标、趋势等度量你的安全性

基线的实例包括：

- 前一周对端口 443 的访问次数
- 用户每天登录服务器 X 次
- 每天使用 su 命令的时间
- 防火墙新端口访问计数
- 每天与环境中服务器联系的主机数量

为了建立基线，我们需要许多规范化形式的数据以及专家的反馈。专家反馈对于辨明正常和不正常的情况是必要的。在大部分情况下，基线不需要训练数据，基线的关键假设如下：

- 有可用的数据
- 过去不是灾难性的，也就是说，在我们计划获得基线的数据中没有缺口等
- 基线是一个正确的模型，它对不规则 / 随机现象无效（这会导致不好的基线）

基本的思路是，你计算一段时间的基线，以便了解正常操作的情况。然后，一旦你收集足够的数据点，就可以开始使用技术，了解何时发生基线之外的事件，从而预示可能的问题。为了改进你的基线，最好是收集尽可能多的数据。拥有的数据越多，结果就越准确。

最大的问题是，要收集多少数据，才能说你已经掌握了标准的情况？理想状况下，你应该收集 4 周至 6 周的数据点。在某些情况下，可以使用一两周的数据。我们来看看计算基线时使用的基本公式。

首先，我们使用均值方法，找出数据点的平均值。

$$\text{均值} = \frac{\sum_{k=1}^{n} x_k}{n}$$

这是简单地累加所有数据点，然后除以所收集的数据点数量。

$$\text{标准差} = \sqrt{\frac{\sum_{k=1}^{n}(x_k - \mu)^2}{n}}$$

接下来，我们将使用标准偏差公式。标准偏差用于测量数据中的变异量。该公式先求每个数据点减去均值的平方（以去掉负值），然后除以样本中数据点数量，再求取上述结果的平方根。最后，我们必须介绍标准误差公式。

$$\text{标准误差} = \frac{\sigma}{\sqrt{n}}$$

标准误差用于确定置信区间，是标准差除以数据点数量的平方根。换句话说，我们用这个值表示未来值可能落入的区间，在统计上并不一定相关。一般使用 95% 的置信度值（不过有时也使用 90% 和 99%）。这一主题的更深入讨论参见 Kachigan（1986）。

现在我们已经了解了所有基本知识，可以概述计算基线的步骤：

1）获取数据集的均值。

2）计算标准偏差。

3）计算标准误差。

置信区间

正态分布

应该注意的是，置信区间只在数据遵循正态分布（贝尔曲线）时才有效。如果你的数据不遵循这种分布，可以运用几种技术，如将数据转换为某种近似于正态分布的形式。如果你想要学习更多这方面的知识，可以查阅参考文献中的书籍。

4）将你的标准误差乘以 1.96（基于 95% 置信度，深入的研究参见 Kachigan(1986)）。有些教科书使用 2 而不是 1.96。

5）加上或者减去从第 4 步中得到的数值，定义你的 95% 置信区间。你使用这个区间来比较未来的数据点，如果未来的数值落入这个范围，它们和基线不一定有太大差异。如果它们落在区间之外，可能在统计学上与基线有差异。

6）在基线上添加新数据并去掉过时的老数据。这很关键，因为旧数据往往会使结果产生偏差。你还需要新的数据来发现行为中的变化。

图 10.1 展示了基线的生命周期。

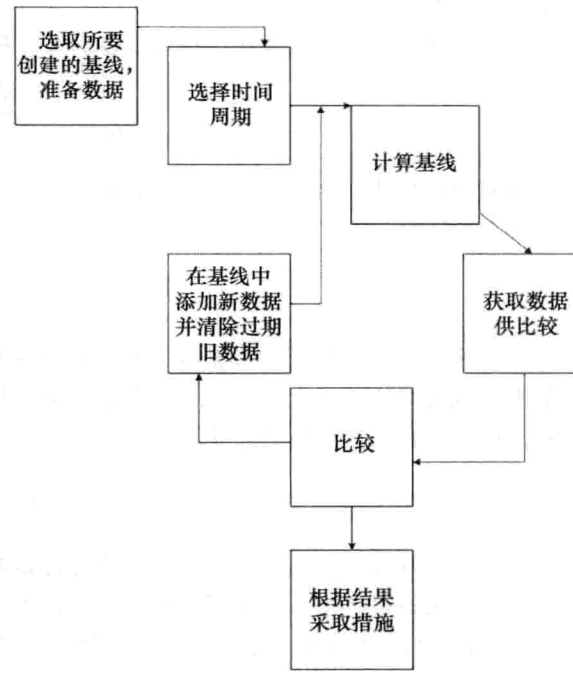


图 10.1　基线生命周期

你应该对哪类事物应用基线，换言之，哪种基线更好？在日志数据的世界里，有一些需要观测的关键事件。我们在这里列出一些：

- 用户登录和注销数量，不管成功还是失败。
- 入站和出站网络流量（字节数）。
- 特定端口 / 服务 / 协议的网络流量。
- 管理账户使用 / 访问。
- 运行在服务器上的进程。
- 硬件健康统计——取决于特定硬件供应商支持的测量，可以关注错误率（硬件错误率）并尝试确定特定硬件部件失效。
- 域名系统（DNS）请求。
- 动态主机配置协议（DHCP）请求。
- 发送和接收的日志数据总数，这可用于检测你的日志数据源出现异常情况或者遭受攻击的情况。
- 新的攻击类型。
- 新的用户名。
- 每天按照传感器统计的日志消息类型。
- 按照协议和端口统计的日志消息类型。
- 日志消息类型（新）。
- 每天按照传感器统计的协议。
- 每个来源独特警报计数。
- 每个来源独特端口计数。

哪些事件不适合于作为基线？

- 在你的环境中，从日志角度看与你所监控的指标无关的随机事件。
- 独自变化的事物，例如对服务器上一个文档的访问。
- 只在某些时候显得重要的大偏差。

现在，我们将深入讨论一些使用基线的话题。

10.3.1 阈值

阈值用于确定何时某些指标超过了基线的值。例如，假定你知道特定的服务器正常情况下每小时平均接受10次失败的登录。如果突然发现这个值跃升到23，那么你就应该展开调查。这个例子还说明了时间对阈值的影响。阈值在与自由时间间隔一起使用时通常没有作用，随着时间的推移，常规的活动最终会超过任何阈值，这就是在阈值上绑定时间有助于确定异常事件的原因。但是在某些情况下，你所绑定的时间可能相当长。例如，如果你希望捕获慢速的攻击，就需要在很广的时间范围（例如数周或者一个月）中布网。而且，除此之外，唯一性是此时的关键。和X个不同的日期相比，你可能对访问的IP更感兴趣。

阈值的自动化创建可能很实用。但是如何进行自动化的阈值创建呢？这一问题的解决方法之一是查看24个小时周期内的日志数据（这时你确实需要规范化日志数据）。据此，你可以执行如下分析：

1）收集在最近24小时内没有发现的日志数据/事件计数。

2）收集在上一周内没有发现的日志数据/事件计数。

根据你收集到的计数，可以看到异常情况是什么样子的。这将为你提供可用的基本阈值，特别是在你的分析说明这些事件来源是未知的情况下。你还可以使用同样的逻辑研究出现在DMZ中的目标IP地址等。

10.3.2 异常检测

异常检测处理前所未见的事件的检测。按照前面的描述，基线的使用是异常检测的一种形式。这种说法没有错，因为我们将跟踪置信区间。异常检测究竟是什么样子的？考虑一下拒绝服务（DoS）攻击。如果有人试图攻击你的基础设施，使它们无法使用，你可以使用统计基线，知道基础设施接收的连接何时增加，并且可能在DoS成为问题之前检测出来。

10.3.3 开窗

假定我们希望能够确定用户在正常观测时间之外登录，或者用户在下班时间异乎寻常地从家中登录。你首先需要做的一件事情是跟踪环境中每个用户每天的登录时间。然后，你可以建立一个行为基线，并使用前面描述的建立基线步骤的第4步和第5步，确定是否有值得调查的事件。例如，如果Bob几乎从不在晚上9点和次日上午8点之间登录，当他突然这么做时，你的基线将帮助你检测出这个事件，以便进一步调查。

另一种技术较为简单，将与基线窗口一端的距离除以窗口的大小。可以使用一个较小的随机阈值确定是否发生了与模式（这里是登录时间）的偏差。更多这方面的内容可以参见《Singer and Bird》(2004)。

开窗非常有用，但是必须小心从事。例如，简单地跟踪某人在何时登录是很好的做法，但是这并没有考虑病假、节假日安排，假日等。上述变量称为季节性参数，对此的讨论超出了本书的范围。几乎所有统计方面深入讨论的书籍都会详细介绍这一主题，包括Kachigan（1986）。

10.4 机器学习

机器学习（Machine learning，ML）是人工智能（Artiicial Intelligence，AI）的一个分支。ML 使用算法和统计方法创建能够学习的系统。许多 ML 算法的基本概念和建立基线相同。用数据和统计方法相结合，从过去的经验和行为中学习，完成模式识别、预测系统、欺诈检测、语音识别等任务。ML 算法基本上分为两类：监督方法和非监督方法。在监督学习中，有一位教师负责更正算法犯的错误。在非监督方法中没有教师。这类算法从可用数据中学习，试图识别数据集中存在的关系。

对 ML 的详细讨论超出了本章的范围。更详细的内容参见本章最后参考文献中的一些书籍。

对于本书的主题——日志数据，我们可以谈及一类可能有用的 ML 算法。这就是 k- 最近邻（k-Nearest Neighbor，kNN）技术。

10.4.1 kNN 算法

什么是 kNN 技术？这是一种 ML 算法，它的目标实际上是用一组预先发现的模式分类未知的模式。我们感兴趣的是发现未作标记的数据中的模式。这种技术的关键是用于确定与未知模式最相似的已知模式的距离（Distance），它的核心就是距离的计算。常用的计算方法是欧几里德距离公式。

$$距离=\sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

kNN 的基本前提是定义不同模式的特征。每种特征表现模式的某种独特性。你拥有的模式和特征越多，就越有可能检测到未知的模式。预先定义模式，你就更有可能检测未知模式。那么，这个距离公式起什么作用呢？该公式使用两个数据向量；p 和 q。其中一个向量是已知模式的集合，另一个是我们试图检测的模式。当我们在每组向量和每个模式的特征中循环时，计算每个已知特征和未知特征的距离。k 起作用的地方是我们取得 k 个最近邻，并使用多数投票方案。这样做的好处是减少误分类，因为可能影响投票结果的例子较多。

我们用一个例子来了解 kNN 在实践中的应用。

10.4.2 将 kNN 算法应用到日志

下面列举了我们用于例子的一组特征：

1）过多的出站流量：EOT。

2）过多的入站流量：EIT。

3）下班时间 VPN 登录：VPNLI。

4）防火墙接受：FWA。

5）防火墙拒绝：FWD。

6）从内部网络之外登录：LOIN。

7）连续多次失败登录：MFL。

8）至少 1 次成功登录：SL。

9）单一来源探查多个目标 IP：SSPMD。

10）单一来源探查多个目标 IP 和端口：SSPMDP。

表 10.1 展示了一组模式和每个模式拥有的特征集。空白列意味着模式没有该特征。

表 10.1 模式和特征

模 式	EOT	EIT	VPNLI	FWA	FWD	LOIN	MFL	SL	SSPMD	SSPMDP
可能的暴力破解登录				Yes			Yes	Yes		
可能的暴力破解登录				Yes			Yes	Yes	Yes	
端口扫描			Yes						Yes	Yes
端口扫描									Yes	Yes
可能的渗漏	Yes		Yes	Yes						
可能的渗漏	Yes		Yes			Yes				

应该注意的是，我们尝试的是分类和检测需要多个事件的场景，以便组成更高级的行为，也就是说，我们并不是简单地分类单一日志消息，而是用多个消息定义模式。

如果我们应用上一小节中描述的 kNN 算法，就可以生成某些结果。表 10.2 展示了几个未知模式通过 kNN 算法的结果。

表 10.2 未知模式 kNN 识别结果

EOT	EIT	VPNLI	FWA	FWD	LOIN	MFL	SL	SSPMD	SSPMDP	结 果
	Yes				Yes	Yes	Yes			可能的暴力破解登录
								Yes		端口扫描
									Yes	端口扫描
Yes										可能的渗漏尝试
Yes		Yes								可能的渗漏尝试

未知模式的分类还不错。选择 kNN 是因为它容易理解和实现。简单的 Google 搜索就能找到 Java、C 和 Perl 等实现。使用 kNN 的一个缺点是，如果特征集和未知集较大，可能花费较多的计算时间。它还需要许多内存来进行处理。

10.5 结合统计分析和基于规则的关联

本章介绍的最后一个主题是统计分析和基于规则的关联。更确切地说，我们希望关注这两种方法如何结合起来，使得各自的概念更加强大。在关于 kNN 算法的小节中，该算法的

结果与基于规则的过程很类似，唯一的例外是 kNN 算法可以用部分数据作出决策。SIEM 供应商总是在规则处理组件之外还使用统计方法。图 10.2 展示了嵌入统计数字的关联概要逻辑流程。

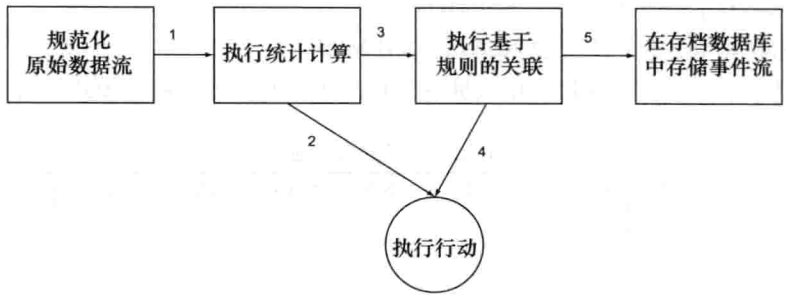


图 10.2　统计和基于规则关联的概念流程

图中展示的是原始日志消息通往最终存储位置的端到端流程。统计计算在这里起了重要的作用。不管你打算购买商业化的日志分析或者 SIEM 平台，还是自行构建，都要记住一些要点：

- 系统的统计部分和规则引擎的集成情况如何？有些平台创建的所有关联规则都包含了条件语句，这些条件以统计计算的形式表达。其他平台则允许在规则关联前后进行统计分析，反之亦然。
- 扩展 / 增强平台的统计组件是否简单？你应该能够调整分析组件，而不需要统计学的学位。
- 统计分析是否有适应性？思路很简单，它是否使用了某种 ML 或者数据挖掘技术，帮助你应对不断变化的黑客威胁。

10.6　小结

本章介绍了一些最基本的统计分析方法。虽然基本，但是它们在应用到日志数据分析时非常有力。在理解正常行为时，基线至关重要。然后，我们应用了阈值和窗口等技术，作为基线的特殊应用，寻找可能不太正常的行为。机器学习是统计分析的一个有趣的领域，可以监督或者非监督的方式使用。建议你学习更多的统计学和机器学习的概念。

参考文献

Kachigan, S. K. (1986). *Statistical analysis: An interdisciplinary introduction to univariate & multivariate methods*. New York: Radius.

Singer, A., & Bird, T. (2004). *Building a logging infrastructure*. USENIX Association.

第 11 章 Chapter 11

日志数据挖掘

本章要点

- 数据挖掘简介
- 日志挖掘简介
- 日志挖掘需求
- 我们要挖掘什么?
- 深入感兴趣的领域

11.1 概述

大部分日志分析技术都要求分析人员对要在日志中搜索的内容有某些特定的了解。例如，他可以“浏览”服务器日志，找出“已知的坏日志条目”（就像 OSSEC 所做的那样，该工具的信息参见第 15 章）。这些日志条目代表着攻击、利用、服务器故障，或者使用字符串匹配或者正则表达式可以发现的其他违规行为。观测这些条目需要大量的领域知识；在这种情况下，从审核数据到运行查询和搜索直到解读结果，根据结论采取行动的整个日志分析过程中，都需要安全和特定日志类型的专业知识。换句话说，你必须知道所要问的问题，才能得到答案，这是件棘手的事。此外，即便是启动这项任务也需要耐心，因为人们可能长时间地查看日志，却一无所获，日志中甚至可能完全没有需要的内容。

在本章中，我们将描述一些方法，它们能够发现日志文件中与安全相关的有趣模式，而不需要确切地知道我们所要找的，从而也就不需要在所有分析阶段上的“耐心”，也不需要

“专业性需求”。我们将研究这些方法的一些实用结果，演示工具，并讨论如何将其用在维护安全性和 IT 运营可用性以及辅助依从性活动过程中发生的各种场景。

因为我们将要介绍的技术与数据挖掘有许多相似之处，我们必须先简单地为不熟悉的读者们介绍一下数据挖掘。

11.2 数据挖掘简介

我们将从数据挖掘（Data Mining，DM）的“词典”定义开始，这一定义可以在大不列颠百科全书的在线版本上找到（Encyclopedia Britannica Online，http://www.britannica.com/EBchecked/topic/1056150/data-mining）：数据挖掘，在数据库和计算机科学中也称为知识发现，是发现大量数据中有趣以及有用模式和关系的过程。该领域结合了统计和人工智能的工具（例如神经元网络和机器学习）以及数据库管理，以分析大的数据集。

对于我们的目的，其他两个定义也很有用。它们是：“从数据中提取隐含的、过去未知和可能有用的信息”[㊀] 和“从数据库或大数据集中提取有用信息的科学”[㊁]。另一个对我们的目的有用的定义是：“以发现数据库中隐藏的事实为目标的信息提取活动”。

注意这些定义中的共同点：

- 数据挖掘（以下简称 DM）处理一个数据池，可能非常大。
- 这些数据在计算机系统上以机器可读的格式存在（例如关系数据库系统）；换言之，它是结构化数据。[㊂]
- 可用数据可能导致我们做出一些有趣的结论，但是也可能无法得出结论。
- 我们试图获得一些有用的结果，这需要搜索或者分析数据。

用上述项目构造 DM 展现了它和日志分析之间明确的相似性。我们有一个很大的日志记录池，需要理解生成这些日志的计算机系统试图告诉我们什么。它们想要描述的有时显而易见，但是在其他情况下隐藏在大量无意义或者不相关的信息中。因此，日志分析和 DM 之间的相似性使我们确信，DM 对于我们手上的挑战——弄清日志的含义可能很有用。

让我们更深入地讨论数据挖掘。数据挖掘的领域涵盖广泛的科学方法、软件应用和使用场景。在此，我们试图为你展示数据挖掘与安全日志分析相关的部分。DM 当然也在从零售业到欺诈分析及风险管理等许多商业领域中发挥着作用。

一般来说，有两种数据挖掘方法：预测性方法和描述性方法。每个人都匆忙地尝试用预测性 DM 来窥探未来，而描述性方法同样有用，因为它能够揭示适用于过去和现在的隐藏真

㊀ Frawley,Piatetsky-Shapiro,and Matheus(1992)

㊁ Hand,Mannila,and Smyth (2001)

㊂ 无结构数据上的数据挖掘被称作“文本挖掘”，今天的许多日志更类似于无结构文本数据，因此文本挖掘技术很实用。尽管作者在这一领域进行了积极的研究，但是本章将专注于结构化数据挖掘，而不是文本挖掘

相，这本身就是很有用的特征。描述性 DM 还能够以发人深省的方式描述数据集，从而几乎“解释”了发生的情况。另一方面，预测性 DM 可以根据已有的数据预报尚不可用的数据。

因为 DM 在零售业有很多应用，我们在转向日志挖掘之前的 DM 示例也将来自这个领域。描述性 DM 可以揭示，如果你购买一个汽车 CD 播放器，还可能购买其他哪些物品（例如碟片、电缆、音箱），从而帮助商店将它们放在播放器的附近。另一方面，预测型 DM 可能帮助商店找出你下一次购买的物品（或者何时将因为购买太多物品而破产……）

单独的 DM 技术很多，表 11.1 展示了一些较常用的技术。

表 11.1　常用的数据挖掘技术

类　型	技　术	示　例
描述性	集群	组合具备共同爱好的人
描述性	关联规则发现	发现数据中与其他人相关的参数
描述性	常见项目集发现	寻找出现在不同组别的对象
描述性	序列模式发现	寻找以特定序列出现的对象
预测性	分类	将数据分为各种类别
预测性	回归	预测数据集与正在挖掘的类似的行为
预测性	偏差检测	了解数据点是否与之前挖掘的模式相符

数据挖掘过程通常被描述为如下步骤：

1）获取数据相关领域的*主题专业知识*——“知道你在数据挖掘中将要做什么”很关键，这和其他领域相同（甚至更重要）。DM 工具没有“魔力”，除非用户了解所要挖掘的数据，否则它可能没有结果。[⊖]

2）*定义目标*是显而易见的关键步骤。虽然探测技术已知，我们仍然需要知道想要收集、准备和挖掘数据的原因，以及用这些结果来做什么。

3）规划收集，然后*收集数据*——这为未来的 DM 工作打下基础。数据有增大到超出控制的危险趋势，就像我们在审计日志数据中所观察到的那样。

4）为了使 DM 过程高效（甚至仅仅为了它能够进行），往往需要进行*数据预处理和清理*。DM 专家估计，这一阶段的工作量占据整个挖掘过程的 60%。这个阶段通常涉及处理信息丢失的部分以及可能重复的数据点，对“噪声”数据进行平滑、识别和更正数据集中的其他不一致问题。有时候，这个步骤也被称为“数据净化”。

5）*数据整理和转换*包括抛弃过大的数据，以及应用旨在通过删除数据集的某些部分，使数据更容易管理的算法。

6）下一步是*选择所要应用的方法*。我们希望进行数据分类，或者找出序列？我们要制作基线，还是发现相关规则？所选择的方法可能由前几步定义的目标驱动。

⊖ 这一“限制”杀死了许多有价值的数据挖掘项目……

7）这样就完成了吗？不，我们需要选择实现上面所选方法的具体算法。研究人员和工具供应商不断地开发出新的算法，并努力改进现有算法的模式和性能。正确的选择可能使你在 10 分钟内高效地完成任务，反之则可能遭致灾难——花费数天的时间，却无法正确地利用数据库。

8）运行挖掘软件，获得结果。这可能指的是看到数据的图形（如果结果很直观）或者表格以及其他文本表现形式。

9）现在，正如一些人所估计的那样，困难的部分开始了：找出结果的真正含义。从简单的文本报告到花哨的可视化形式，这一阶段是我们真正可以理解前面的所有步骤是否挖出“金矿”的时候。

我们可以看到，上述的步骤和常规的日志分析过程很相符。这就是我们在下一小节开始说明如何使用 DM 类技术进行日志分析的原因。

让我们将焦点放在最后一步，讨论关于“兴趣度”的一个有趣问题。[⊖] 如果 DM 成功且返回的结果正确（意味着方法学和采用的应用程序没有缺陷），如何保证我们对这些发现感兴趣呢？

例如，本书的作者之一在日志数据上试验相关规则发现，提出了一些 DM 方面的发现：如果连接的协议是 DNS（域名系统），很有可能发送的端口是 53。这一发现很可能得到轻蔑的回应，因为虽然挖掘算法确实完成了自己的工作，但是我们对这一结果毫不关心。对于分析人员来说，很明显几乎所有 DNS 通信都会发生在端口 53 上。这说明 DM 技术能够发现“隐藏”的模式，但是，为什么我们能够确定用户关心这些发现呢？“兴趣度”这一指标是研究人员对此进行量化的一个尝试。对它的另一个实用定义为：“如果一个模式很容易被人们理解，在新数据或者测试数据上验证时有某种程度的确定性，可能实用、新颖或者验证了用户试图确认的假设，那么它就是有趣的。”这个度量有些时候被分类为客观兴趣度或者主观兴趣度。前者由计算机使用任何“兴趣度”算法确定，而后者由人类专家定义。这种区别明显对日志分析有用，因为许多人认为它“是一门艺术，而不是科学”，而分析人员是任何技术实用性的最终裁判。

我们在这个阶段强调“兴趣度”的主要原因是，在研究数据挖掘时，人们往往期待奇迹。例如，“我把日志放入系统中，它将告诉我现在需要知道的所有事情。”这种事情很难发生，因为即使系统有经过完美调整的数据挖掘能力，它对于你现在所感兴趣的事情也仍然一无所知。但是有一个好消息：DM 比其他任何技术都更可能接近于上述的目标。

在这段简介之后，我们已经做好准备，了解数据挖掘技术是如何应用在我们的日志分析中的。

⊖ 是的，这是一个数据挖掘术语。它在 ftp://ftp.gte.com/pub/kdd/kddterms.html#Interest 被确切地定义为“模式实例质量的度量，具有多个维度。主维度是样本集合上的验证、全局可靠性、与其他已知模式实例相关的冗余度、普遍性、简洁性和实用性。”

11.3 日志数据挖掘简介

虽然日志数据挖掘从技术上讲是一种分析，但是我们希望说明，它从常规的日志分析结束的地方开始，从而和前者有所区别。此外，作者不喜欢“日志异常检测”的说法，因为它往往（例如在信息安全的学术性出版物上）是“这不起作用，但是如果有一天几颗卫星连成一线，并且为某位神仙献上合适的祭品，它就可能起作用”的另一种说法。

了解了数据挖掘方法和技术的一些知识之后，我们退后一步，研究我们对日志数据应用数据挖掘所要实现的目标。一般来说，我们寻求的是：

- 改进日志分析的质量，提供更好的答案和预测能力。
- 提出不需要少有且昂贵的专业知识的高级和有效方法。

Richard Bejtlich 在他的《Tao of Security Monitoring》一书中描述了对网络安全监控的看法。这种监控明显涵盖了日志（例如来自入侵检测或者预防系统的日志）以及网络流量信息，例如 netflow 和完整的包捕获信息[⊖]。他的 NSM 方法的主要信条是，具备高度技能的分析人员遵循精心优化的分析过程，并运行工具以便从输入数据（如日志、警报和数据包转储）中获得深入的信息。这一方法有一个小问题：这种分析人员并不容易找到，至少可以说，他们的成本对于手上的工作来说是难以承受的。因此，整个方法论轰然崩塌，因为它依赖于训练有素、理解能力强和积极的分析人员（甚至“专家”），要求他们始终坐在控制台前。因此，这种方法只能用于资金雄厚且安全性至关重要的组织中。

通过使用数据挖掘和其他高级自动化分析方法，我们试图将大部分“分析的负担”（或者“荣耀”，这取决于你提问的对象）转移到软件和自动化系统上，远离具备超常技能（因此也就难以找到）的分析人员。至少，我们更希望这样的分析只在过程的早期阶段（正如前一小节所述，在我们设计特定挖掘方法的时候）中是关键。在这种情况下，专家可以定义挖掘过程的细节，然后让技能水平较低的操作人员运行算法，并对结果采取行动，而不会降低效率。

让我们来简单地看看 DM 技术如何应用到日志，为什么这么做，以及这么做的好处。我们所说的“日志挖掘”，其含义就是“数据挖掘技术在高级日志分析中的应用”。

我们将首先提出“为什么挖掘日志”这个问题，在理想的状况下，我们的愿景是减少所需的人为工作量，提升自动化系统在日志分析中的角色重要性。简而言之，日志挖掘是实现上述目标的方法之一。实际上，正如我们在前面所提到的，它可能是最有希望的方法。所以，日志分析面对如下的挑战：

- *过多的数据*：日志数据的体积很大，超出了分析系统和分析人员的能力，从而摧毁了获得答案的可能性。确实，日志可能达到数 G 字节，并扩大到数 T 字节；所以你需要工具来应对这种“洪泛”。

⊖ Cisco 的 netflow 格式用于从基础设施设备收集网络流量信息（更多信息参见《Cisco IOS Netfow》：http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html。

- 没有足够的数据：因为各种不同的原因，数据的关键部分缺失，使得日志分析超出了应有的难度。
- 各种各样的记录：需要分析太多不同和不相似的日志来源，才能得到真相。这个问题是因为缺乏统一的审计标准；大部分应用程序的日志采用创建者开发的格式，从而导致大量分析上的挑战。MITRE 的“通用事件表达式”（Common Event Expression，CEE 参见 http://cee.mitre.org）等未来标准将会解决这个问题。
- 假警报：日志中充满了不能以任何可测量方式反映现实的消息（网络 IDS“假阳性”大都处于这个类别）。
- 重复数据：不同的日志引用相同事件，没有任何指示；这种情况往往因为不同日志源缺乏时间同步而变得更加复杂。
- 难以获得数据：虽然一个设备可能以专有格式创建很有用的日志数据，但是由于缺乏“标准”[⊖]日志格式（如 syslog 或者 Windows NT 事件日志）而无法与集中日志收集“协作”。类似地，获得详细的主机审计日志也可能是一个挑战。

已经有许多技术被开发出来，以应对上述挑战。那么，如果我们有了这些其他方法，为什么还要挖掘日志呢？下面是主要的原因：

- 通过启用更类似于人类的模式识别，我们只需要在分析的早期阶段具备较高的专业知识，从而减少对高技能的分析人员的依赖。
- 处理其他较为常规的方法无法有效分析的稀疏数据。对几个 G 甚至几个 T 的数据进行查阅的任务最好是留给全自动的方法，如日志挖掘。
- 检测其他方法无法发现的隐藏事实。是的，日志挖掘能够提高检测入侵痕迹的效率。
- 将结论生成的负担转移到机器上；这样，目前仅能由人类完成的任务也可以自动化。人们仍然根据结论采取行动，但是他们不需要为了试图了解所发生的情况而绞尽脑汁。
- 尝试预测问题，而不是寻找处理已发生情况的方法。虽然 DM 不提供对这类预测的保证，但是确实比其他方法更加接近这一目标。

我们应该注意，尽管数据挖掘有“替代”人类的趋势，但是算法总是应该由专家定义和调整。不过，DM 技术明显地降低了操作人员的技能需求。

在作者所曾经进过的一些早期实施中，基于 DM 的日志分析系统的用户评论道，在相对简单的调整过程之后，这些系统变得“比初级分析人员更强”。在这种场景下，人们采用使用 DM 类日志分析方法的系统，将它们部署在整个企业中，查看来自不同冲突检测和防火墙设备的详细审计数据。系统开发人员和高级分析人员在初始阶段中使用这些系统；他们还训练曾经只会看着控制台上的日志记录流动、执行初始警报调查的初级分析人员。现在，有了经过调整的基于 DM 日志分析系统，分析人员只专注于调查，因为系统在选取重要警报和事

⊖ 通过对本书的阅读，你已经知道这些“标准”并不是真正的标准，它们只是许多系统使用的常见格式。

件方面和那些分析人员做得一样好。

在对日志数据应用数据挖掘之前，让我们先来看看必要的条件。

11.4　日志数据挖掘需求

预先要注意的是，日志数据挖掘的许多需求和任何重要的日志分析一样。不过，有些附加因素可能使日志数据更适合于挖掘，或者从可选需求变成强制性的需求；

1）数据集中化：对于过滤和总结等常规日志分析来说，只专注一个位置是很好的，而对于日志挖掘这一点变得很关键，因为挖掘算法比任何分析人员处理的数据都要多得多。

2）规范化：如上所述，查看集中化的不同数据源需要统一的信息格式，这种格式并不是真正的标准，只是表现日志数据的统一方式。

3）关系存储：关系数据存储（如 RDMBS）在此不可或缺，但是如果只进行简单的分析和过滤，这一点可以忽略。[⊖]

这种规范化通过搜索日志中的公共字段实现。常见的字段包括：

- 时间
- 来源
- 目标
- 协议
- 端口
- 用户名
- 事件 / 攻击类型
- 交换字节数
- 其他

因此，规范化和集中化数据可能受制于日志数据挖掘算法。现在，我们已经做好准备，研究应用数据挖掘所要寻找的东西。

11.5　挖掘什么

我们已经确定，DM 方法在我们不确定所要搜索的内容时最有用（否则，我们就可以过滤或者搜索所需要的内容）。正如我们前一小节所提示的，我们如何找到“有趣”的信息？如上所述，有趣的信息可能包括用户所没有预想到的和可以采取行动的信息。

系统管理员和安全分析人员觉得“有趣”和有帮助的发现有哪些常见的例子呢？

- 受感染的系统传播恶意软件：在许多情况下很明显，每个安全管理员对受感染后在企

⊖ 上述文本挖掘当然不需要这里提到的两种需求。文本挖掘方法只在少数日志数据上进行。

业范围甚至互联网范围内传播的系统都有着很大的兴趣。尽管有防病毒软件和其他专用解决方案，日志挖掘在跟踪这种危害性极大的系统方面已经证明很有价值。

- 遭到入侵的系统：每个获得认证的安全专家都应该对了解攻击者或者他们的走狗（恶意软件）已经接管网络中的一个或者多个系统感兴趣。
- 成功的攻击：如果攻击者刚刚成功地在你的系统上打开缺口，婉转地说，知道这一点可能“有趣”；虽然这和前一条相关，但是它通常指的是攻击的早期阶段，这时攻击从尝试发展为成熟的入侵和对系统的利用。
- 内部人员违规使用和知识产权窃取：怀有恶意的黑客和蠕虫出尽了风头，而内部网络违规使用相比之下似乎显得简单。但是，内部人员拥有这一王国的所有钥匙，可能造成更加严重的破坏；更糟糕的是，检测他们的攻击比检测普通的恶意软件困难得多。内部攻击（即使失败）的清晰证据无疑是管理员和分析人员感兴趣的信息。
- 隐蔽通道 / 隐藏后门通信：除非你“精于此道”，否则你的网络中可能不会常用到隐蔽通道；因此，网络安全管理人员很可能对了解这一情况感兴趣。
- 探查增加：虽然大部分敏感的政府网络现在只将互联网探查活动视为噪声，但是如果在日志中反映出这类活动增加，可以看做攻击的前兆，因此也是有趣的。
- 系统崩溃：虽然“拒绝服务检测”总是遭到嘲笑（从定义上说，这种检测是通过注意到你恰好没有任何服务来进行的），但是系统管理员可能并没有监控所有系统的正常运行。

因此，上述总结展示了一些我们希望通过挖掘日志发现的有趣事实。

对计算机来说，“有趣”的准则很难定义，但是用日志挖掘可以做到！让我们再来看看“无趣”的常见例子，这些例子不符合上述的“兴趣度”准则——意外和可采取行动：

- 探查（不是意外的）：网络探查和扫描总是在发生，人们越来越能预料到它们。人们应该了解探查，但是不可能花费资源去寻找它们。与此同时，这类探查数量（每天、每周、每月等）上的变化更可能成为有趣的信息。
- 常见的失败攻击（不是意外的）：如果你的安全架构很稳固，你就能够预期到失败的攻击；这些攻击的发生有各种原因（攻击者的技能是其中之一）。和探查类似，人们应该了解它们，但是不会花费资源去寻找它们。
- 正常消息（不是意外的）：日志充满了表明某些例行过程完成和其他完全正常的事件的消息，这些记录是用于审计和其他目的的，它们明显不出人意料，我们也不会去挖掘它们。但是，变化仍然很重要：常规的消息停止出现，或者开始表现出较高或者较低的频率，这可能令人感兴趣。
- 被阻止的攻击（无法采取行动）：这和失败的攻击类似，如果你的安全措施阻止了攻击，即使这种攻击是你所没有想到的有趣攻击，也不需要立刻采取行动。确实，可能有必要进行调查，但是这仍然与“兴趣度”准则不符。
- 系统状态更新（无法采取行动）：类似于正常事件，这些事件无法诉诸行动。同时，系统状态更新在某些不寻常的时候发生可能很令人感兴趣。

下面我们更详细地通过日志挖掘寻找有趣的事物。

11.6　深入感兴趣的领域

让我们来回顾一下数据挖掘的预演，并再次复习介绍过的方法，看看如何将它们应用到手上的任务。下面是我们可以挖掘，并且有希望找到有趣信息的内容：

- 罕见的事物：由于某种原因，这并不常发生！如果发现之前没有发生过的事件，它很可能是恶意的。因此，罕见事件是我们挖掘的主要候选。罕见的攻击、罕见的系统消息、几乎从不登录的用户，都是值得检测的！
- 不同的事物：虽然我们不提倡“日志恐惧症”，但是应该注意和以往不同的事物。这是数据挖掘的基线方法起作用的地方。
- “不相称”的事物：虽然和上一类紧密相关，但表现得“不相称”的日志记录必须加以挖掘，因为它们很可能包含有趣的信息。
- 古怪的事物：如果某些事物显得古怪，可能是出现麻烦的象征。这和上面两个类别有微妙的差别。要了解古怪的事情（如 DNS 系统连接到 packetstorm.com，自动下载攻击工具），不需要任何基线和预先的数据收集工作。
- 向不寻常的方向发展的事物：分析人员和管理员可能发现，有些日志记录完全改变了正常的通信方向。正如上例中所看到的，到你的服务器的连接完全合法，但是服务器向外连接（即使使用的是无害的端口，如 TCP 端口 80）至少应该引起重视。
- 最常见的事件：虽然属于总结和报告的范畴，严格说不属于数据挖掘，但是“X 大事件”对于寻找有趣的日志记录及其模式仍然有用。毕竟，如果排名有所上升（例如：传输带宽最主要用途），它仍然是有趣的。
- 最不常见的事物：类似于罕见事务（简单地说，“发生次数最少的事情”），这是“X 大事件”的“邪恶姐妹”，甚至比前者更有用。最不吸引人的系统往往成为未来攻击和破坏的跳板。
- 无趣事物的奇怪组合：这粗略地遵循下面的公式：“善”+“善”=“邪恶”。是的，一组完全正常的日志记录也可能组成危险的事件。最简单的例子是端口扫描，它看上去只是一组（往往数量很大）无害的连接请求。
- 其他无趣事物的计数：某些无趣事物的计数可能令人感兴趣。而且，这种计数的变化往往更加重要。ICMP“ping”数据包的突然增多（即使在防火墙之外）可能意味着拒绝服务，尤其是这种 ICMP 洪泛在没有预警的情况下突然发生时。

我们举个例子来说明。在这个例子中，发现了一个黑客事故，该事故涉及漏洞扫描器的未授权使用。我们都知道，大部分公司都部署网络 IDS（也称作“NIDS”），但是许多公司并不理解采购这些设备的价值。这种情况发生的原因包括日志中的大量假阳性，它们削弱了人们对这些系统的信任。基于特征的系统尤为严重，这种系统标记具有特定模式的数据包和连

接。为了实现 NIDS 的价值，我们可以使用日志挖掘方法，旨在从常规的噪声和假警告中标记出真正的攻击。注意，在这种情况下我们可能并不知道攻击是否成功，只是了解攻击者的存在和重点，并将攻击与错误触发 NIDS 的无害事件区分开来。

当网络基础设施组件或者被误用的检测系统（如 IDS）记录合法连接时，它们通常会在日志中生成少数独特的事件类型。例如，通过防火墙的连接生成一个连接消息。即使扫描防火墙也可能为每个连接会话生成一个事件，以及一个用于整个扫描的事件类型（connection_denied）（这个事件将重复许多次，但是类型都一样）。类似地，“假阳性”通常不与同一批主机之间的其他可疑活动（例如侦察性扫描或者其他攻击）相关。相反，假警告更可能完全独立发生或者以大量互不相关的同类日志记录类型的形式出现。这里，我们所指的“会话”是如下要素的独特组合：

- 源
- 目标
- 协议
- 源端口
- 目标端口

因此，如果我们按照会话组织数据库中收集的事件，并记录每个会话中独特事件的数量（往往和攻击类型的数据相同），我们就有办法区分真正的攻击和合法流量及假警报。这种日志挖掘方法使用一种通用模式，以获得潜在破坏性攻击的特定及有趣结果。同样，我们不能找出攻击成功的标志，只是鉴定它的真假。

11.7 小结

本章简单地介绍了数据挖掘方法在实用日志分析上的应用。我们阐述了日志挖掘和日志知识发现是审核日志数据的一种新颖方法，在实际工作中真的有效！许多日志类型可能从某种挖掘中获得有用的结果。虽然数据挖掘是一个复杂的领域，但是日志挖掘（至少其中的某些方法）并不那么困难，可以在许多环境中实现。而且，在需要高技能分析人员和很长时间的常见分析方法无效时，日志挖掘能够提供帮助。

最后，该领域中的积极研究，包括通过文本挖掘进行的无结构日志挖掘，可能为人们带来更加实用的方法。

参考文献

Frawley, W., Piatetsky-Shapiro, G., & Matheus, C. (1992). Knowledge discovery in databases: An overview. *AI Magazine*, 213–228.

Hand, D., Mannila, H., & Smyth, P. (2001). *Principles of data mining*. Cambridge, MA: MIT Press. ISBN: 0-262-08290-X.

第 12 章 Chapter 12

报告和总结

本章要点

- 定义最佳报告
- 身份认证和授权报告
- 更改报告
- 网络活动报告
- 资源访问报告
- 恶意软件活动报告
- 关键错误和故障报告

12.1 概述

任何必须查看日志数据的人都知道，只查看原始日志没有什么作用。但是，大部分日志分析最终都涉及查看一定时期内特定系统集的日志数据总结——报告。我们必须分析的数据越多，就必须进行越多的总结和聚合。实际上，大部分人审核日志的第一次经验可能发生在对日志数据报告的审核上——宝贵的“10 大用户”、“报告期内最常见的攻击”以及其他报告。

确实，总结和报告是日志分析的支柱：根据日志生成的按带宽排列的主要连接、按国家排列的主要攻击、身份认证失败最多的用户和其他报告，是大部分人考虑日志分析时最常想到的。商业化的日志分析和 SIEM 工具也更多地查看报告而非原始日志数据。

总结很有效，并减少了你需要查看的数据量，但是在总结中我们杀死了数据，这有可能

丢失有用的信息；此外，数据可能因为我们选用的总结类型而被丢失。例如，如果你关注 10 大用户，不要忘记排名倒数前 10 的用户可能也很有趣。

最后，报告通常相当多，选择哪一种报告是个挑战。举个例子，某个供应商曾经宣传他们的产品有“13 000 种报告”——似乎这是一件好事…

在此基础之上，许多组织苦于多种依从性框架（PCI DSS、HIPAA/HITECH、FISMA 和本书其他章节中讨论过的许多其他框架）以及高级的威胁（恶意软件、邪恶的黑客、移动威胁、云安全挑战等）。而且，怀有恶意的内部人员现在更有可能对企业造成损害或者欺诈。与此同时，信息技术对企业和政府组织的重要性已经有了很大的提高，并将继续提高。

12.2 定义最佳报告

报告在日志分析中用途广泛，因此报告的可能种类很多，重要的是创建一个“最佳报告”列表，或者至少列出行业中适用性最广的报告。

本章就是创建这一列表的一次尝试。报告被组织为 6 个宽泛的类别（或者报告类型），每个类别都有适用于大部分组织的具体示例。它们的设计是与技术无关的，可以用于商业化、开源或者内部开发的日志管理及分析工具生成，也可以使用更高级的 SIEM 工具。

最重要的报告类别如下：

1）身份认证和授权报告。

2）系统和数据变更报告。

3）网络活动报告。

4）资源访问报告。

5）恶意软件活动报告。

6）故障和关键错误报告。

在本章的其余部分，我们将用具体的例子介绍各个类别。

12.3 身份认证和授权报告

这些报告识别以各种不同权限级别（身份认证）访问各种系统的企图（无论是成功还是失败），以及特定权限用户的活动，以及使用特权功能（授权）的企图。

重要性

身份认证是控制现代系统访问权限的主要关卡和手段。从简单的密码到令牌以及加密机制，审核整个组织的身份认证活动是关键的安全性活动之一。

具体报告

这个类别的关键报告有：

- 按照用户、系统、业务单位列出的所有失败和成功登录：这可能是一个或者多个显示不同系统、访问方法（本地或远程）和用户的成功及失败登录的报告。这种报告的价值在于，它要求你不能仅仅记录成功或者失败的登录，而是对两者都要记录。
- 禁用 / 服务 / 不存在 / 默认 / 挂起账户的登录企图（成功，失败）：这种报告覆盖对不应该访问的账户和服务的访问企图。不管失败还是成功，对安全专业人士来说都是有趣的。
- 下班时间 / “休息”时间的所有登录：和上面的报告类似，这种活动通常在访问成功时特别令人感兴趣。但是，对这种事件必须加以调查，尤其是在系统管理员全天候工作的环境中。
- 用户在试图登录的不同系统中身份认证失败的计数：这种聚合报告检测账户扫描，在这种扫描中，单一机器检查许多系统上的同一个账户或者不同账户，和老式的“主机扫描”有些类似。
- VPN 身份认证和其他远程访问登录（成功，失败）：所有登录企图在合适的场合下都是令人感兴趣的，而通过 VPN 或者其他远程连接方法进行的远程登录尝试就更为有趣，应该小心跟踪。
- 特权账户访问（成功，失败）：根用户或者管理员登录、su 命令的使用、Run As 命令使用以及其他平台和系统上的等价操作都必须加以考虑，因为特权用户造成的破坏一般比常规用户严重得多。

多次登录失败之后，同一个账户登录成功：这种报告需要基于规则（SIEM 风格）的关联来生成，但是跟踪多次失败后的成功连接明显令人感兴趣，因为它几乎总是说明有人企图猜测登录证书（来源：http://chuvakin.blogspot.com/2010/08/updated-with-community-feedback-sans_06.html）。

谁能使用这些报告

这些报告有很普遍的适用性，具体取决于系统的覆盖范围。首席安全官（CSO）可以审核整个组织的身份认证摘要，安全分析人员可能在日常日志审核中使用这些报告，事故响应人员可能在事故调查时使用这些报告，而系统管理员可能在自己的系统上使用这些报告。

表 12.1 展示了系统登录企图。

表 12.1 展示登录企图的报告示例

系 统	账户名称	源 IP	状 态	方 法	计 数
Venus	administrator	10.1.1.2	失败	本地	1
Jupiter	anton	10.11.12.13	成功	本地	1
Mercury	root	10.1.2.3	失败	SSH	893765

12.4 变更报告

这些报告识别各种系统和安全的关键变更——配置文件、账户、监管和敏感数据以及系统或者应用程序的其他组件。

重要性

对信息系统的未授权变更导致许多代价沉重的崩溃、数据丢失和安全事故。除此之外，攻击者往往修改你的系统，以便在未来能够访问。勤勉地跟踪变更还将改进你的整体 IT 运营状况。

具体报告

这个类别的关键报告有：

- 用户和组的添加 / 变更 / 删除：攻击者常常添加新的账户，然后在访问之后删除。这些活动应该在授权方式下进行。
- 管理员 / 特权组中账户的添加：对管理员账户和其他特权用户的变更应该在账户变更的跟踪列表中排在首位。
- 密码变更和重置——由用户进行以及由管理员进行：密码变更通常和新账户的创建一样重要。这些操作可以由用户进行，也可以由管理员进行。此外，这种报告可以用于确定授权的密码变更是否按照策略的安排进行。
- 网络服务的添加 / 变更 / 删除：允许网络连接的新服务可能向附加的攻击开放你的网络，它们常常是攻击者进行的。
- 对系统文件（二进制文件、配置）的变更：对系统文件（如二进制文件和配置文件）的变更，不管是偶然、计划还是恶意的，都需要小心跟踪。
- 对其他关键文件的变更：除了二进制可执行文件和配置文件之外，各种系统可能有不同的关键文件，对这些文件的访问也要跟踪。
- 文件访问权限的变更：危险变更的变种之一是文件权限的变更；如果没有考虑到这种变更，它们可能造成敏感数据的丢失。
- 对敏感文件的变更：或者对敏感文档的下载 / 复制。
- 系统、应用程序、用户进行的应用程序安装和更新（成功、失败）：各系统的所有应用程序安装和更新必须记录日志，至少，这些日志在事故响应中很有用。

谁能使用这些报告

这些报告有很普遍的适用性，具体取决于系统的覆盖范围。首席安全官（CSO）可以查看整个组织的变更摘要，安全分析人员可能在日常日志审核中使用这些报告，事故响应人员可能在事故调查时使用这些报告，而系统管理员可能在自己的系统上使用这些报告。

表 12.2 展示了 Linux 系统上的所有账户和组添加。

表 12.2　展示账户和组添加的报告示例

日　期	系　统	账户名称	操　作	对　象	状　态
1/10/1111:11AM PST	Venus	root	添加账户	anton	成功
1/11/11 11:11AM PST	Jupiter	anton	添加组	sudoers	成功
1/10/11 11:11AM PST	Venus	root	添加账户	root1	失败

12.5 网络活动报告

这些报告识别各个系统的可疑和有潜在危险的网络活动，以及需要为常见法规跟踪的活动。

重要性

网络是信息资产威胁的主要来源。显然，网络也是从现代组织中窃取信息资产的主要途径。

具体报告

这一类别的关键报告有：

- 按照系统、连接数量、用户、带宽、独特目标计数列出的所有来自内部和 DMZ 系统的出站连接：对环境中的出站连接的信息有多种切片方式，但是主要的精华都一样：跟踪从你的网络连接到外界的用户，是检测入侵和破坏以及恶意软件（还有滥用网络权限的用户）的方法。
- 在下班时间内部和 DMZ 系统的所有出站连接：使用防火墙和 Web 代理日志，人们可以得到上述报告更有针对性的版本，只跟踪非正常时段内的出站访问。
- 传输的最大文件（入站、出站）或者传输字节数最大的会话：这两种报告都使组织能够跟踪露骨的数据窃取和带宽滥用。
- 向外部网站上传 Web 文件：根据代理日志，人们可以跟踪被上传到外部网站的文件，以及作为 Web 邮件附件的文件。
- 按照内容类型（exe、dll、scr、upx 等）和协议（HTTP、IM 等）分类的所有文件下载：跟踪从 Web 进入环境的文件也很重要，可以通过跟踪不同协议和方法的文件来完成。
- 使用许多不同协议 / 端口的内部系统：虽然没有可靠的方法能够知道来自合法内部系统的恶意软件活动，但是突然开始通过许多新端口和协议“交流”已经被看做恶意活动的一个征兆。
- 最常成为多种 NIDS、NIPS 或者 WAF 警报源的内部系统：最有用的报告之一是跟踪生成许多不同类型的警报，像“圣诞树”一样的内部信息资产。
- 按照用户名称、会话总字节数、会话计数、内部资源使用量分类的 VPN 网络活动：在上一小节中，我们重点介绍了跟踪 VPN 登录的必要性，但是 VPN 的使用情况也应该跟踪，以便找出 VPN 访问和流量中的异常状况。
- 内部系统使用 P2P ：虽然用户破坏 AUP 可能是这一工作的重点，但是 P2P 软件也涉及意外和恶意的数据窃取和丢失。
- 无线网络活动：无限网络设备能够记录许多不同的事件，将它们作为前述的 VPN 和其他远程网络访问机制进行跟踪也很有用（包括用户名和 Windows 名）；有关无线数据的另一个有用的报告包括流氓 AP 检测和流氓 AP 关联日志。
- 一段时期内的日志容量趋势：虽然这严格上不能算是网络活动报告，但是审核网络产

生的日志容量极其有用，它可以作为日志数据的整体指标。

谁能使用这些报告

这些报告有很普遍的适用性，具体取决于系统的覆盖范围。首席安全官（CSO）可以查看整个组织的网络活动摘要，安全分析人员可能在日常日志审核中使用这些报告，事故响应人员可能在事故调查时使用这些报告，而系统管理员可能在自己的系统上使用这些报告。

表 12.3 展示了一个组织网络上所有 VPN 账户访问和活动的报告。

表 12.3　VPN 账户访问与活动

日　期	VPN	用户名	系　统	操　作	状　态	计　数
1/11/11	VPN1	anton	antonlaptop	登录	成功	2
1/12/11	VPN1	antom	antonlaptop	登录	失败	1
1/13/11	VPN2	root	Lapt19847	登录	失败	77

12.6　资源访问报告

这些报告识别整个组织各种系统、应用程序和数据库资源的访问模式，可以用于活动审计、趋势分析和事故检测。

重要性

资源访问的跟踪可以用于揭示内部人员违规使用甚至欺诈。它们在事故响应期间很有价值，可以用于确定攻击者访问和可能破坏或者修改的资源（参见 12.4 小节）。此外，资源访问可以用于安全以外的目的，比如容量规划和其他用途。

具体报告

这一类别的关键报告有：

- 下班时间访问关键系统上的资源：类似于上面所提到的下班时间的网络访问和登录，这种报告可以用于跟踪在不寻常的时间在关键和受监管的系统上的访问和活动。
- 被代理阻止访问禁用网站次数最多的内部用户、恶意来源等：这种多用途 Web 访问报告可以用于从跟踪受害系统到数据泄露跟踪以及改进生产力等多种目的。
- 文件，网络共享或者资源访问（成功、失败）：这种报告只能用于特定的被审计资源；启用文件访问日志（Windows）和系统调用日志（Unix）总是会导致数据溢出。
- 数据库主要用户：为了用于安全活动跟踪，这种报告必须排除已知的应用程序对数据库的访问；理想情况下，用户或者开发人员不能直接访问生产数据库。
- 查询类型摘要：同样，排除已知的应用程序查询，能够将这种报告转化为异常检测工具，展示异常的数据库访问。

- 所有特权数据库用户访问：和服务器及应用程序一样，所有特权用户活动应该加以记录并定期分析。
- 执行 Insert、Delete 数据库命令的所有用户：除了跟踪应用程序和用户访问之外，单独跟踪可能毁坏数据的破坏性命令也很有意义；在这里，排除已知应用程序的查询是实用的方法。
- 执行 Create、Grant 命令，更改数据库模式的所有用户：除了跟踪应用程序和用户访问之外，单独跟踪可能毁坏数据和更改数据库实例本身的破坏性命令也很有意义。
- 数据库备份摘要：备份为从数据库提取大量数据从而实施数据窃取提供了一种清晰的方法；这种报告使你能够审核旧数据库的备份，捕获未经授权进行备份的人。
- 向外发送附件最多的内部电子邮件地址：在许多电子邮件访问报告中，这种报告引人注目，它对于检测和调查内部人员的违规使用和数据窃取起着关键的作用。
- 所有电子邮件附件内容的类型、大小和名称：类似于上一种报告，这种报告可用于跟踪信息泄露，检测用户通过邮件发送敏感信息。
- 所有发送电子邮件的内部系统（排除已知的电子邮件服务器）：寻找环境中被垃圾邮件发送程序感染的系统的一种简单方法。
- 日志访问摘要：记录日志，然后审核对日志的访问是法规的规定；这一基本报告应该排除你自己对日志数据的查看。

谁能使用这些报告

这些报告有很普遍的适用性，具体取决于系统的覆盖范围。首席安全官（CSO）可以查看整个组织的资源访问摘要，安全分析人员可能在日常日志审核中使用这些报告，事故响应人员可能在事故调查时使用这些报告。资源所有者可能使用这些报告规划容量，做出其他业务决策。

表 12.4 展示了多个服务器的所有文件访问的报告。

表 12.4 多个服务器的文件访问

日 期	服务器	用户名	文件名	访问类型	状 态	计 数
1/11/11	Win1	anton	Expenses.xlsx	读	成功	1
1/12/11	Win2	anton	Roadmap.ppt	读	成功	1
1/13/11	NFS	anton	Blank.docx	写	失败	37

12.7 恶意软件活动报告

这些报告总结各种恶意软件活动和与恶意软件可能相关的事件。

重要性

各种形式的恶意软件仍然是当今大型和小型组织所面临的主要威胁。虽然防病毒工具近年

来已经加入了阻止恶意软件的功能，但是仍然必须使用其他信息来源（如日志）抵御恶意软件。

具体报告

这个类别的关键报告有：

- 恶意软件检测趋势及结果：恶意软件检测的摘要或者趋势报告，还展示系统和结果（已经清除或者保留），这是一个好的起点。
- 来自防病毒工具的仅检测事件：所有防恶意软件工具记录检测到恶意软件但没有清除（原因各异）的情况；这种记录下来的“保留”能够帮助许多组织避免大规模的破坏。
- 所有防病毒保护故障：考虑到当今的恶意软件已经具备了与防病毒工具周旋的能力，所有的死机、检测引擎卸载、更新失败等都必须记入日志并进行审核。
- 指向已知恶意 IP 地址的内部连接：人们可以使用这个极其有用的报告，利用日志（如防火墙或者其他设备的日志）和公开的 IP 地址黑名单，这种简单的方法能够避免组织中有价值的数据落入恶意软件运营者之手。
- 最不常见的恶意软件类型：和其他“后 10 位”（和“前 10 位”相对）报告一样，这个报告很好地揭示了组织中不寻常从而可能具有破坏性的恶意软件。

谁能使用这些报告

这些报告对于所有安全专业人士都有用，涵盖了从负责桌面防病毒的初级管理员到负责整个组织安全的 CSO 的范围。这种报告对于事故响应和恶意软件感染调查也有用。

表 12.5 展示了整个网络和为期一周的日志数据中的病毒类型，按照计数的升序排列。

表 12.5　网络中的病毒类型

恶意软件类型	状态	受感染系统计数
VirusX	检测	1
VirusY	检测	1
Botz	隔离	2

12.8　关键错误和故障报告

这些报告总结各种严重的错误和故障指示，这些指示往往在安全上有直接意义。

重要性

错误和故障日志消息往往提供安全威胁的早期指示，包括安全专用设备（如 IDS 和 IPS 系统）所没有捕捉到的高级威胁。注意不寻常的错误消息，在网络中出现可能具备破坏性的新威胁因素时常常能得到回报。

具体报告

这一类别的关键报告有：

- 按照系统、应用程序、业务单位排列的关键错误。虽然在表面上没有安全意义，但是出现在日志中的各种错误消息（尤其是第一次出现）应该加以调查，因为它们往往提供恶意活动的早期指示；下一小节中的分析性报告可以非常有效地用在这类日志消息的管理中。
- 系统和应用程序崩溃、关闭、重启：每当应用程序崩溃，由于失败的攻击或者其他原因，业务功能可能受到影响，不能将这些事件只当成可用性问题，而应该进行调查，找出攻击的非直接早期指示。
- 备份故障是影响业务持续性并可能影响监管依从性的关键事件，此外，未授权备份（在这里指的是失败的尝试）可能是因为攻击者试图窃取数据而触发的。
- 内存、磁盘、CPU 和其他系统资源容量耗尽 / 达到限值：资源的高利用率往往是因为攻击者或者业务系统未授权使用所引起的，也可能是因为洪泛攻击、拒绝服务或者暴力破解攻击。

谁能使用这些报告

这些报告对于所有安全专业人员都有用，涵盖了从负责桌面防病毒的初级管理员到负责整个组织安全性的 CSO 的范围。这种报告在事故响应和恶意软件感染调查中也很有用。其他 IT 人员也能从中获益。

表 12.6 展示了 Unix/Linux 服务器池中磁盘满和 CPU 高利用率消息的报告示例。

表 12.6 磁盘满和 CPU 高利用率报告

服务器	事件类型	日期
Serv1	磁盘满	10/1/11
Sirius	磁盘满	1/1/11
VenusX	CPU 负载 100%	1/2/11

12.9 小结

正如我们在引言中所说，基于所有日志数据创建每个人都能使用的报告列表是不可能的。不管依从性法规如何强制，创建这种列表的所有尝试都注定会失败。但是，拥有几乎每个人都应该使用的报告列表，并根据情况进行调整是很实用的做法。

Chapter 13

第 13 章

日志数据可视化

本章要点

- 视觉关联
- 实时可视化
- 树图
- 日志数据合成
- 传统日志数据图表

13.1 概述

本章简单介绍日志数据可视化的主题，对此主题的全面研究超出了本书的范围。Conti（2007）和 Marty（2009a）对这一主题进行了深入的介绍，我们鼓励你研读这些书籍和所有其他参考文献。

13.2 视觉关联

回忆我们在第 7 章和第 8 章所讨论的日志分析技术。在那些章节中，我们明显利用了人类的眼睛，观察和寻找日志中的异常表现。如果你的日志数据较少，视觉关联就很有用处。你首先必须开始将日志数据收集到一个聚合点。下一步是使用工具查看日志数据文件中的输入事件。有时候，你可以使用 UNIX tail 命令，特别是在所有的日志消息收集到一个文件的

情况下。如果你需要观察超过一个日志文件，这一任务就变得复杂了。

这种技术的缺点是，你实际上每秒钟只能进行大约 5 个左右事件的视觉关联。这意味着，你最终必须花费很多时间观察日志数据，以非实时的方式工作，这挫败了关联的主要目的：实时寻找信息。

13.3　实时可视化

当然，有替代的方法。SIEM 平台的大部分日志分析工具都有某种实时查看事件的界面。图 13.1 展示了 OSSIM 事件查看器[⊖]。

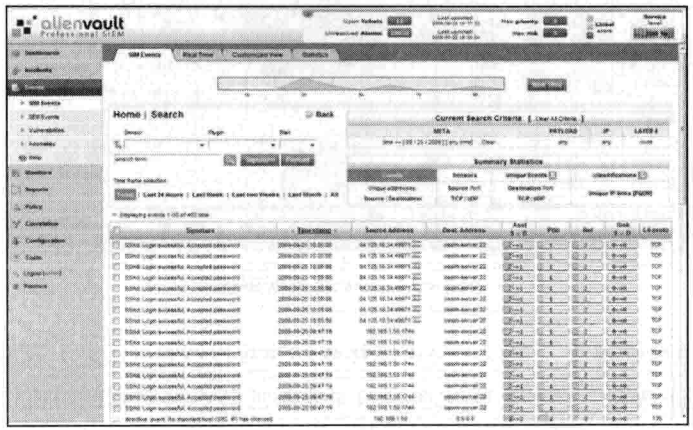


图 13.1　OSSIM 实时事件查看器

这类工具使你可以在规范化的日志数据（事件）从远程日志收集点流入你的分析服务器时，直观地进行检查。而且，大部分这类工具允许你执行过滤、关联和突出显示感兴趣事件等操作。这种界面很重要，原因如下：

1）可以聚合离散的事件数据。

2）多个用户可以同时使用该工具，从而提高了可伸缩性。

3）更容易找出未知或者之前从未见过的日志数据。

4）更容易实现特征数据的重新分类。

13.4　树图

树图（Treemap）是一种可视化技术，层次化的数据用不同的颜色显示，代表可变的叶节点。这一过程可以更好地标识模式。

⊖ http://communities.alienvault.com

图 13.2（取自 Marty 2009b）以树图方式显示了防火墙数据。图 13.2 中的每个方框顶部有一个代表源的 IP 地址。然后，每个方框中包含的所有 IP 地址是每个源连接的目标。例如，192.141.69.45 是源 IP，而子方框（如 192.134.0.，192.26.92 等）是 192.141.69.45 联系的目标。

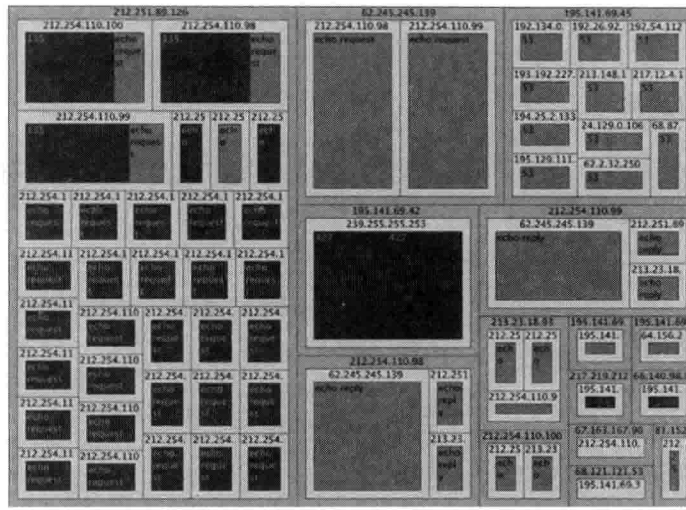


图 13.2　防火墙数据树图

从黑白图片中很难辨认，实际上防火墙接受的连接以绿色（浅色）表示，防火墙丢弃的连接以红色（深色）表示。这种可视化在检测需要添加防火墙规则的时机、验证规则是否有效时很有用，甚至可以将防火墙日志与 IDS 日志结合，创建一个交叉的树图。

13.5　日志数据合成

网络安全行业专家 Jimmy Alderson 使用 GraphViz，创造了日志数据的可视化表现形式。他发明了 Constellations（星座，合成）这个术语。GraphViz 是通过图表和网络表示信息的一种工具，其核心发布版可以在 http://www.graphviz.org/ 上找到。这里将介绍 GraphViz Perl 模块，该模块可以在 http://search.cpan.org/CPAN/authors/id/R/RS/RSAVAGE/GraphViz-2.10.tgz 上找到。你必须安装 GraphViz 软件包，才能安装和使用 GraphViz Perl 模块，该模块需要其他 Perl 模块，它将会告诉你遗漏了什么。

图 13.3 是展示 GraphViz 作用的一个示例。

现在你知道这个名称的由来了。在黑白图形中可能不太明显，但是这种图确实是有颜色的。它的思路是将基础设施中的各种服务或者服务器表现为节点。因为其他系统都与这些服务相联系，GraphViz 可以将这种通信表现为源系统到目标系统的一条边。

图 13.4 展示了在浏览器中使用的 GraphViz 图。

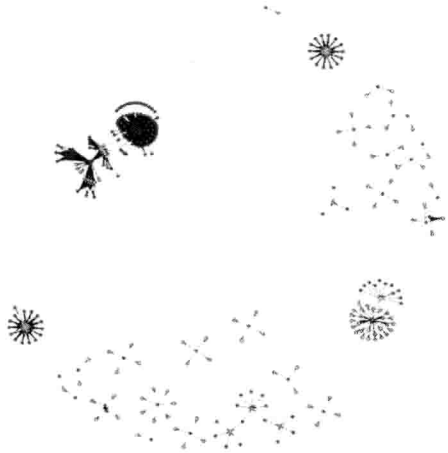


图 13.3　GraphViz 图示例

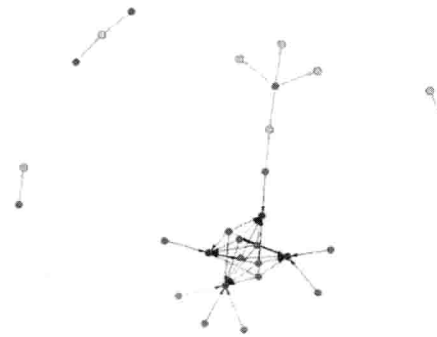


图 13.4　可点击的 GraphViz 图

为了展示更多细节，浏览器本身被忽略。图 13.4 中的图设计为可点击图像。图中每个节点都是某种网络通信的目标或者源。本质上，图 13.4 是由节点和边形成的有向图。如你所见，每个节点到另一个节点都有一条边，边上有表示通信方向的箭头。下面的 Perl 脚本不仅创建可点击的图，还包括一个可用于浏览该图的 HTML 文件：

```
#!/usr/bin/perl
#
# Author: Jimmy Alderson
#
use GraphViz::Small;
my%dst_hash = ();
my%edge_hash = ();

my $g = GraphViz::Small->new(layout => 'neato', bgcolor => 'black');
while ($line = <STDIN>){
   chomp($line);
   undef($sip);
   undef($dst);
   undef($count);
   if(($sip,$dst,$count) = $line =~ m/^([^ ]+) ([^ ]+) (\d+)/)
   {
      ($port) = $dst =~ m/:(\d+)/;
      ($dip) = $dst =~ m/(\d+):/;
      if(!exists($sip_hash{$sip}))
      {
         $g->add_node("$sip", URL =>  "$sip.html", tooltip =>
```

```
            "$sip", style => 'filled', fillcolor => 'red',);
            $sip_hash{$sip} = 1;
         }
         else
         {
            $sip_hash{$sip}++;
         }
         if(!exists($dst_hash{$dst}))
         {
            $color = get_fill_color($port);
            $color2 = get_border_color($port);
            $g->add_node(name =>"$dst", URL => "$dst.html",
            tooltip => "$dst", style => 'filled', color =>
      "$color2", fillcolor => "$color", cluster => "$port",    rank
=> "$port");
            $dst_hash{$dst} = 1;
         }
         else
         {
            $dst_hash{$dst}++;
         }
         $edge_hash{"$sip-$count#$dst"} = 1;
      }
   }
   foreach $edge (keys%edge_hash)
   {
      if(($s, $c, $d) = $edge =~ m/([^-]+)-(\d+)\#(.+)/)
      {
         ($port) = $d =~ m/:(\d+)/;
         ($dip) = $d =~ m/(\d+):/;
         $color = get_fill_color($port);
         $g->add_edge("$s" => "$d", color => "$color", minlen => "$c");
      }
   }
   open(FH, ">map.html");
   print FH '<HTML><BODY BGCOLOR=black><img src="map.png"
   usemap="#map"><map name="map">';
   print FH $g->as_cmap;
   print FH '</map></body></html>';
   close(FH);
```

```
print $g->as_png;
sub get_fill_color
{
   my $port = shift;
   my $control = 'magenta';
   my $web = 'white';
   my $email = 'yellow';
   my $encrypted = 'green';
   my $ftp = 'pink';
   my $netbios = 'blue';
   my $console = 'cyan';
   my $proxy = 'orange';
   my $db = 'green';
   my $suspect = 'red';
   my $chat = $suspect;
   my%colors = ();
   $colors{'0'} = $control;
   $colors{'21'} = $ftp;
   $colors{'20'} = $ftp;
   $colors{'23'} = $console;
   $colors{'25'} = $email;
   $colors{'53'} = '#00FF00';;
   $colors{'80'} = $web;
   $colors{'443'} = $web;
   $colors{'135'} = $netbios;
   $colors{'139'} = $netbios;
   $colors{'161'} = $control;
   $colors{'1433'} = $db;
   $colors{'1434'} = $db;
   $colors{'1080'} = $proxy;
   $colors{'5190'} = $chat;
   $colors{'8080'} = $proxy;
   $colors{'6000'} = $console;
   $colors{'6667'} = $chat;
   $colors{'31337'} = 'gold';
   if(exists($colors{$port}))
   {
      return $colors{$port};
   }
   return 'lightblue';
```

```
}
sub get_border_color
{
   my $port = shift;
   my $encrypted = 'green';
   my $unencrypted = 'yellow';
   my%colors = ();
   $colors{'0'} = $unencrypted;
   $colors{'22'} = $encrypted;
   $colors{'23'} = $unencrypted;
   $colors{'25'} = $unencrypted;
   $colors{'53'} = $unencrypted;
   $colors{'80'} = $unencrypted;
   $colors{'443'} = $encrypted;
   $colors{'135'} = $unencrypted;
   $colors{'139'} = $unencrypted;
   $colors{'1433'} = $unencrypted;
   $colors{'5190'} = $unencrypted;
   $colors{'6000'} = $unencrypted;
   $colors{'6667'} = $unencrypted;
   if(exists($colors{$port}))
   {
      return $colors{$port};
   }
}
```

上述脚本预期的输入数据格式如下：

```
source_ip destination_ip:port count
```

source_ip 是指向 destination_ip 指定端口的通信源。Count 是 source_ip 和 destination_ip 在指定端口上通信的次数。这种格式要求你对日志数据进行某种预处理。防火墙和 IDS 事件最可能成为这种图表的选择，但是你也可以对任何数据制作图表。关于上述脚本，需要注意的是，它根据目标端口底层服务是否加密以及服务的类型（Web、数据库等），使用不同颜色来表示节点间的边。

上述脚本的运行示例如下：

```
$ ./graph_bloom.pl < data > map.png
```

我们将上面的脚本命名为 graph_bloom.pl，data 文件包含了以前面提到的格式处理过的数据。输出文件有两个：map.html 和 map.png，我们在命令行中指定。这种性质的图表可能不适合于实时使用，但是对于接近实时的分析或者取证分析可能有用。这一工具解决的主要

问题是创建了查看和分析日志数据的另一个维度。这些合成的有趣特征之一是不同类型的流量以独特和可辨识的模式显示出来。例如，HTTP、SMTP、DNS 和 Netbios 各有自己的合成特征，训练有素的人很容易发现异常。

13.6 传统日志数据图表

从可视化流量模式和不同来源与关键服务器的交互角度看，前一小节介绍的图表非常有趣。世界上许多组织使用 Multi Router Traffic Grapher(MRTG) [⊖] 分析路由器接口使用等情况（输入输出字节）以及内存和磁盘使用等操作系统资源。它可以用于收集和制作任何图表。例如，你可以制作防火墙接受和抛弃图表，也可以制作源和目标 IP 地址之类的图表。

下面的 MRTG 配置说明如何执行脚本，返回防火墙接受的连接数量：

```
Target[fw.accepts]: '/usr/bin/perl /usr/local/bin/accepts.pl'
MaxBytes[fw.accepts]: 512
Options[fw.accepts]: gauge
Title[fw.accepts]: Number of firewall accepts
YLegend[fw.accepts]: FW Accepts
PageTop[fw.accepts]: <H1>FW Accepts </H1>
<TABLE>
   <TR><TD>System:</TD> <TD>My Firewall</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>""</TD></TR>
</TABLE>
```

accepts.pl 脚本简单地打开一个数据文件，打印其中的一行。为了使这一过程正常工作，必须预先处理日志数据，这和前一小节所做的类似。MRTG 一般通过 UNIX cron 工具自动定期运行。所以，你要有一个脚本或者程序来预先处理防火墙数据，并将防火墙接受的连接数量转储到一个文件中。下面是 accepts.pl 的一个例子：

```
#!/usr/bin/perl
open(FILE,">processedFile");
while($line = <FILE>)
{
   chomp($line);
   print "$line\n";
}
#
# The following code prints the system uptime and the hostname. These two
# items need to be included in every script that you write and should be
   the
```

⊖ www.mrtg.org

```
# very last thing that is printed.
#
chomp($uptime = '/usr/bin/uptime');
print "$uptime\n";
chomp($hostname = '/bin/hostname');
print "$hostname\n";
```

下面是需要记住的一些要点：

- 因为 MRTG 每五分钟运行一次（你实际上可以更频繁或者更少运行它，但是 5 分钟是很好的采样间隔），你的预处理脚本需要在 5 分钟内完成，以使 MRTG 的下一次运行正常完成。
- 你的预处理脚本应该跟踪原始日志数据文件的位置，以便处理。
- 对于你想要制作图表的每个防火墙，需要一个单独的配置块。

13.7 小结

我们介绍了可视化的几个关键主题。视觉关联可以进行日志数据的 guick-and-dirty 分析，但是可能很费精力，容易出错。实时可视化为日志分析企业提供了更多的宏观视图。树图是直观地发现日志数据中模式的一种有趣方法。日志数据合成为你提供了日志数据的点对点视图。最后，传统日志数据图表采用更加以网络管理为中心的日志数据可视化方法。

参考文献

Conti, G. (2007). *Security data visualization: Graphical techniques for network analysis*. San Francisco: No Starch.

Marty, R. (2009a). *Applied security visualization*. Upper Saddle River, NJ: Addison-Wesley.

Marty, R. (2009b). Data visualization with treemaps—A hands-on tutorial. *Network World*. Network World, May 20. Web. June 17, 2012. <http://www.networkworld.com/community/node/42024>.

第 14 章 Chapter 14

日志法则和日志错误

本章要点

- 日志记录法则
- 日志记录错误

14.1 概述

正如我们在前几章中所描述的，来自信息系统的日志的收集和分析已经有几十年的历史了。因此，出现一些关于应该避免的事项的真理也就不足为怪了。在本章中，我们将揭示这些普遍真理，并称它们为日志法则（可能有些冒昧）。

14.2 日志法则

我们的日志法则将覆盖 IT 数据重要来源的整个处理过程——从收集到分析和决策。但是在开始之前，我们首先要介绍定义普遍日志真理时的灵感来源——日志记录和 IDS 的 Ranum 法则。

下面这些法则是 Marcus Ranum（2012）在 20 世纪 90 年代的一次安全演讲中提到的：

Ranum 日志记录和 IDS 第一法则——收集数据的数量绝不能超过你觉得可能使用的数量。

Ranum 日志记录和 IDS 第二法则——无趣的事情发生的次数是我们所感兴趣的。

Ranum 日志记录和 IDS 第三法则——除了和第一法则冲突的情况之外，尽可能收集所有

数据。

Ranum 日志记录和 IDS 第四法则——如果你没有实时的系统管理员，IDS 的实时性就无关紧要。

（来源：http://ranum.com/security/computer_security/archives/logging-notes.pdf）。

这些法则到今天仍然是正确的，并且得到了许多其他法则的扩展和补充。

14.2.1 法则 1——收集法则

第一条法则与 Marcus Ranum 法则的精神类似："不要收集你从不计划使用的日志数据。"

这一日志记录普遍真理似乎和最近某些供应商鼓吹的"收集 100% 数据"和"没有人们不喜欢的日志"相抵触。实际上，这条法则只是说明，每个记录和留存的日志消息都必须有存在的理由。

实际上，"我可能需要用它来调查一个事故"完全是合理的理由，而"哦…，我猜想可能需要它"就不是合理的理由。

实际上，这条法则不仅适用于日志数据的收集，也适用于生成——"不要记录你从不计划使用的日志。"

14.2.2 法则 2——留存法则

虽然可以收集更多的数据，但是开始时很少用到它们。正如我们在本书中所介绍的，那是日志留存的主题，是一个重要的考虑因素。

"只要觉得有可能用到，就尽可能长时间地留存日志数据，如果有法规规定，还应该保留更长的时间。"

我们在本章中还将进一步地暗示这一点，当我们谈到日志记录的错误时，在根据留存规程清除之后一天就发现需要日志消息的情况并不少见。在可能有用的情况下尽可能长时间保存数据，是这一法则的主题。

另一方面，在应用程序崩溃之后数年，不太可能使用调试消息。虽然，我们可以人为地制造一种情况，让这些消息成为最重要的信息，但是不建议保存长达 7 年的所有消息。

14.2.3 法则 3——监控法则

虽然前面几条法则让人们觉得更多地记录日志和更长时间的留存是明智的做法，但是关于监控的法则却恰恰相反："记录你所能记录的一切（尽可能多），但是只在你必须响应时发出警报（尽可能少）。

确实，当你意识到系统可以很轻易地保存几个 PB 数据，即数万亿条日志消息，而安全响应人员一天只能处理一打问题时，这一监控法则也就很容易理解了。

一万亿：1 000 000 000 000

一打：12

差别很大，对吗？这就是日志记录和监控的差异！

正因为组织必须记录日志，然后才能开始监控，因此两者的方法完全不同。我们在本书的其他地方讨论过，组织应该遵循的原则是“记录所有情况，保存部分情况，监控必需的情况。”

14.2.4　法则 4——可用性法则

“不要花费精力，将你的日志记录或者监控系统的可用性提高到超过业务系统的水平”，这一法则有时候听起来有悖常理。但是，它的唯一含义是，安全监控的可用性需求和业务系统相同，而没有必要超过。

确实，确保日志数据的收集和分析，并使之可以用于未来的调查甚至诉讼很重要。但是，大部分业务领导会告诉你，业务系统的可用性更为重要。

14.2.5　法则 5——安全性法则

如果你理解可用性法则，就会喜欢安全性法则：“不要花费比保护关键业务数据更多的精力去保护你的日志数据”。

这个问题看上去很难，是吗？我们更应该保护哪一个：一个秘密的新战斗机计划（好吧，也许只是一个秘密的客户清单或者一位名人的健康记录），还是一组很好的 syslog 消息？日志确实有价值，有时候甚至被认为是业务记录，但是大部分企业都会指出另一组数据比日志更有价值。

这也是只有少数组织会选择加密日志数据的原因。除了加密带来的运营难题之外，在许多情况下，这种控制明显过分了。有些企业满足于对日志存档进行哈希（Hashing）处理，减少对日志的篡改，而其他企业宁可采用严格的访问控制方式。

14.2.6　法则 6——不断变化法则

日志记录、日志管理和日志分析领域中，唯一不变的是变化。“日志来源、日志类型和日志消息都在变化。”

单独考虑一个已经使用 20 年的遗留应用程序，它不会突然地产生不同的日志。但是，围绕这个应用程序的环境可能产生许多不同和新型的日志消息。在当今这个虚拟化和云计算的年代，组织、收集和分析日志都发生了更多的变化。

因此，组织必须持续地审核日志审核策略、规程和运营任务及系统配置。用文档记录日志收集过程的每一步并且使之保持最新，这很有用，从日志生成开始到最终存储位置，显示它流过的所有点。

14.3　日志错误

组织和个人都可能忽略上述日志法则。但是，有一些错误与法则无关，有些错误起源于

许多年前，但是现在仍然适用，而其他一些错误更现代，与当今的 IT 环境相关。

本节涵盖了各种组织在处理信息系统日志管理和计算机及网络设备生成的其他记录时所犯的典型错误。遗憾的是，尽管这方面的意识正在提高，但这些错误在许多组织中仍很常见。

随着技术的持续传播，计算机在我们的商务和个人生活中扮演着更为重要的角色，日志（计算机所生成的记录）也开始变得越来越重要。因此，犯下这些错误的代价不断增高。20 年前，犯下这个列表中的一个错误只会妨碍故障排除或者其他系统相关活动。近 10 年来，计算机控制了生活中越来越多的领域，这些错误（我们并没有刻意夸张）可能关系到某些人的生命。

从防火墙和入侵预防系统（IPS）到数据库和企业应用程序，再到无线接入点、VOIP 网关和虚拟化平台，日志以越来越快的速度增长。而且，这些日志源的位置也在扩展——从系统到云应用程序，再到移动设备。安全性和其他 IT 组件不仅在数量上增长，而且常常启用了更多的日志记录。这一趋势的例子包括 Linux 系统和 Web 服务器，它们现在所带的日志记录水平有了提高。现在的 Windows 系统也记录了比 10 年前更多的事件——每个事件的记录更加详细。

不管是遗留的还是新建的，传统的还是基于云的，所有系统都会生成丰富的日志、审计跟踪、记录和警报，对此必须始终加以注意。因此，许多公司和政府机构试图建立可重复的日志收集、集中和分析过程及工具。

实际上，处于云计算最前端的公司被淹没在日志中。从规模超群的 Google 和 Facebook 到其他 Web 公司，日志不再以 GB 为单位计算，而是 TB 甚至 PB 量级。日志数据总和，包括安全日志和调试及运营日志，每天达到数 T 的情况已经司空见惯。

但是，在规划和实施日志收集及分析基础设施时，组织常常发现它们没有实现这一系统所承诺的效率，实际上，有时候会注意到，效率并没有提高而是降低了。

我们将从明显的错误开始，很不幸的是，这些错误在有 Sarbanes–Oxley 和 PCI DSS 等法规和强制标准的今天仍然很常见。这些错误会毁掉所有从日志获益的机会。

14.3.1 完全没有日志

头号错误就是完全不记录日志。更有趣的说法是："不记录日志，甚至不知道日志是什么，等到想起来时一切已经太迟了。"

有人会问："怎么会太迟呢，不就是个日志吗？"，没有日志可能导致损失收入（PCI DSS 有明确的日志要求，意味着违规可能导致你的信用卡处理特权被 Visa 或者 Mastercard 集团取消，从而使你停业）、声誉（有人从你的数据库偷走了几个信用卡号，但是媒体报道 4000 万张信用卡都已经失窃，因为你无法提出反证）甚至自由（参见媒体上各种关于 Sarbanes-Oxley 和 HIPAA 的恐怖故事）。

甚至准备得更充足的组织也会碰到这个问题。下面是最近的一个例子，你的 Web 服务器已经启用日志了吗？确实，启用日志是两个流行 Web 服务器（Apache 和 Microsoft IIS）的默

认选项。你的服务器操作系统记录日志消息吗？当然，没有人取消 /var/log/messages。但是你的数据库呢？哎哟！ Oracle 的默认选项是不进行任何数据访问审计记录。MS SQL 也许好些？没有那回事，你同样需要进入系统，启动一个合理级别的审计跟踪生成（更多内容参见我们关于数据库日志的论文）。

为了更详细地说明这个问题，我们来研究一个常见的数据库类型，了解默认可以得到哪些日志。Oracle（实际上其他数据库也是）默认记录最低限度的系统日志，某些 DBA 访问的数据库服务器会记录日志，但不记录任何数据访问日志。

那么，数据访问审计记录、模式和数据变更审计记录和配置变更审计记录在哪里？它们都可以配置（你需要阅读相关文档！），但是默认情况下是关闭的。

实际上，在云计算环境中部署的许多系统不仅缺乏日志记录配置设置，而且往往不需要登录。例如，在本书编著期间，某个流行的云计算提供商的环境中没有任何记录入站和出站连接日志的手段。

因此，为了避免这个错误，人们有时候必须超越默认值，确保所部署的软件和硬件启用某种级别的日志记录。在使用没有日志记录设置的系统时，你的选择只能是放弃使用这种系统、与提供商协商（如果它们是很大的公司，这可能不是好的选择）或者增加其他技术以记录活动和行动。

14.3.2　不查看日志数据

第二种错误是不查看日志数据。确保日志存在，然后收集和存储它们很重要（是的，实际上你可以在必要时查看日志数据，从而获益），但是这只是一种手段，知道环境中发生了什么，并能够对此做出响应，预测以后可能发生的事情，才是真正的目的。因此，一旦使用收集日志技术，就需要有一个持续监控和审核的过程，在必要时与行动和可能的升级挂钩。此外，审核或者监控日志的人应该有足够的信息，才能确定实际的含义和所需的行动。

值得注意的是，有些组织在正确的方向上只走了一半：他们只在重大事故之后（入侵、信息泄露或者神秘的服务器崩溃）才审核日志（如果他们没有犯第一个错误，且真的有需要审核的东西），而没有进行持续的监控和日志审核，理由往往是“缺乏资源”。这为他们带来了日志分析在被动反应上的好处，这很重要，但是却无法实现主动反应的好处——知道何时发生不好的事情，或者事态何时变得更糟。例如，如果你审核日志，就会知道防火墙上激活了故障切换，尽管连接仍然保持，但是这种事故同样值得观察。如果你没有继续观察且网络连接断开，你就不得不依赖日志来调查为什么两个故障切换设备都下线了。事实上，主动查看日志帮助组织更好地实现现有网络、安全和系统基础设施的价值。

由于某些监管压力，强调某些类型组织必须查看日志文件和审计记录也很关键。例如，美国的 HIPAA 法规强制医疗组织建立审计记录和分析程序（但是因为强制措施的缺乏而声名狼藉）。更为极端的情况是，PCI DSS 有日志收集和日志监控及定期审核的条款，强调了日志的收集并不是企业自己的事情。

下面例子说明审核的优先级是根据风险评估（将 DMZ 认定较高的风险）和可用技能（因此，应用程序日志排在后面）排定的。

审核优先级从外部高风险到内部风险，从服务器到桌面，从他们所知道的工具（防火墙和 IDS）到可能重要的工具，但是没有相关技能的应用程序。

1）DMZ NIDS

2）DMZ 防火墙

3）安装应用程序的 DMZ 服务器

4）关键内部服务器

5）其他服务器

6）选择的关键应用程序

7）其他应用程序

这种方法能帮助组织避免被过多的日志弄得不知所措的情况。

14.3.3 保存时间太短

第三个常见错误是保存日志的时间过短。这使得安全或者 IT 运营团队认为自己拥有监控和调查或故障排除所需的所有日志，而在事故发生之后惊恐地发现，由于短视的留存策略，所有日志都没了。事故的发现往往需要很长的时间（特别是内部攻击的情况），有时候在犯罪活动或者违规使用发生数月之后。人们可能在存储硬件上省了一些钱，但是因为监管的惩罚而遭到 10 倍的损失。

如果低成本很重要，解决方案是有时将留存分为两部分：短期在线存储（成本较高）和长期的离线存储（便宜得多）。更好的三层方法也很常见，能够解决前一方法的某些局限性。在这种情况下，短期的在线存储得到近线存储的补充，这种存储中的日志仍然可以访问和搜索。最旧且相关性最低的日志记录被卸载到第 3 层，例如磁带或者 DVD，这种存储较为便宜，且无法选择性地访问必需的日志。

更具体地说，某个金融机构存储可以存储 90 天的在线日志，然后在近线可搜索存储中保存 2 年，最后在磁带上保存 7 年或更久。

值得一提的是，强制的 7 年日志留存需求是杜撰出来的！

有趣的是，保留太长时间的日志也可能是个错误。为什么会这样？首先，我们要提醒自己，日志留存等于日志存储 + 访问存储数据 + 日志销毁。

那么，为什么要销毁日志呢？三个因素使得存储太长时间的日志成为一个错误。

首先，在欧洲很盛行（在美国还没有这么做）的隐私法规往往限制数据收集和留存。是的，在某些国家，保存日志是违法的，在规划日志记录策略时必须加以考虑。

其次，诉讼风险管理。你的法律团队一定会提醒你，收集太多在诉讼案件中会被发现的信息不是好事情。最著名的例子是，拥有某些活动的日志（但是没有经过安全团队的审核！）可能被视为对侵权行为知情。这样的情况会使你的组织陷入法律困境。

最后，系统资源利用率往往驱使你限制存储日志数据的数量。如果你有一组繁忙的防火墙，保留数年的日志也许是不可能的，或者成本极高（你不得不学会 PB 的拼写方法）。

下面是一个留存策略的示例，它能告诉你创建日志留存策略、避免错误的一种方法。

所以，需要学习的主要模型是使用留存策略的 3 个维度：

1）日志来源类型——防火墙、IDS、服务器、桌面等。

2）网络位置——DMZ、地理区域、分支机构等。

3）存储层次——在线（例如，日志管理系统）、近线（原始日志的大型磁盘存储）、离线（磁带或 CD）。

这意味着，你选择一种日志来源类型（比如防火墙），考虑它在网络的部署位置（如 DMZ），然后规划它的存储方式（例如，在快速的在线存储中），这将得到一个保存日志的数字。在上例中，DMZ IDS 日志在线存储 90 天，而同一位置的防火墙日志保存 30 天（因为它们的数量较大）。然后，同样的日志被存储到离线存储（第 2 层），保留 3 年。

选择设备的顺序也很重要，因为这将定义日志捕获和留存的实施优先级。

14.3.4　在收集之前排定优先顺序

第四种错误和日志记录优先顺序相关。虽然人们需要优先级来更好地组织日志分析工作，但是现在常见的一种错误是在收集之前对日志记录排定优先顺序。实际上，连某些“最佳实践”文档也建议只收集“重要的内容”。但是什么是重要内容？上述的指导性文档并没有以实用的方式指定这一点。虽然对这一问题有某些解决方法，但是据我们所知，这些方法都可能带来安全漏洞，甚至削弱监管依从性工作。

例如，许多人可能声称网络入侵检测和预防日志天生比其他日志（如 VPN 集中器日志）更重要。在外部威胁与内部违规使用相比处于统治地位的情况下，这种说法可能是正确的。在进行有关信息泄露或者恶意软件感染的内部调查时，最有可能需要 VPN 日志和服务器及工作站日志。因此，对于其他日志类型的重要性可能有类似的争论，这会导致我们痛苦地发现，必须收集所有的日志。

但是你能这么做吗？在回答之前，尝试回答一个问题：你在看到日志之前，能够确定哪个日志更重要吗？这样，上述问题就不再是不能解决的了。为了避免犯这个错误，我们将采用如下的简化措施。

首先，记录一切并不意味着分析人员将审核每个日志记录，而是意味着如果有必要，就可以访问调查和审计人员所需的所有日志记录。

1）记录所有日志。

2）留存大部分日志。

3）分析足够的日志。

4）在一个子集上总结和报告。

5）监控某些日志。

6）对少数日志采取行动。

简而言之，信息的精简发生在日志收集和存储之后，而不是之前。这将使你能够在未来的任何情况下使用日志——不管是用在监管还是安全相关的场合。

14.3.5 忽略应用程序日志

下一种错误是忽略来自应用程序的日志，只将重点放在边缘和内部网络设备，以及服务器，而不进入“较高的层次”去查看应用程序日志。

企业应用程序的范围包括从 SAP 和 Pepople Softs 到处理许多企业关键过程的小型自制应用程序。运行在主机和中型系统上的遗留应用程序到处可见，它们往往也承担着核心业务的处理。日志的可用性和质量在不同应用程序上各不相同，可能完全没有（许多自制应用程序都是如此），也可能极端详细和巨大（许多主机应用程序的情况）。软件开发人员缺乏公共的日志记录标准和日志记录原则，给应用日志造成了许多挑战。

实际上，当应用程序部署在云中，或者组织利用软件即服务（SaaS）云模型时，应用程序日志是监控活动和攻击的唯一手段。注意一个事实：应用程序日志的越来越重要，它们早晚会成为分析需求的一部分。

尽管有上述的挑战，人们仍然必须确保应用程序日志的收集，使其可用于分析并长期留存。这可以通过配置你的日志管理软件收集它们，以及建立一个日志审核策略（用于事故时审核和定期主动日志审核）来实现。

14.3.6 只搜索已知的不良条目

即使最为先进和成熟的组织也可能落入第 6 种错误的陷阱。这种问题是一种隐患，可能严重地降低日志分析项目的价值。它发生在组织只查看日志中已知的问题的情况下。确实，有大量的开源和一些商业化工具被用来过滤和寻找有问题的日志行、攻击特征、关键事件等。例如，“swatch”是一个经典的免费日志分析工具，它只在一件事上很强大：寻找日志文件中已经定义的不良内容。而且，当人们谈到日志分析时，他们通常讲的就是在日志中进行筛选，寻找值得注意的内容。

但是，为了完全实现日志数据的价值，人们必须将其提高到日志挖掘的水平：在对“我们要寻找的信息”没有预先的想法的情况下，发现日志文件中的有趣内容。这看上去很明显，我们怎么可能预先知道所有可能发生的恶意行为，但是这往往被忽视。有时候，列出所有已知的正常事件，然后寻找其他的信息可能更简单。这听起来是一个解决方案，但是这种任务不仅繁重，而且徒劳无功：列出发生在一个系统或者网络上的所有正常事件，可能比列出所有不正常事件更麻烦。

不同事件和不正常的功能或者行为很多，只凭列出所有可能性来找出攻击的痕迹，这并没有效果，需要更加智能的方法！有些数据挖掘（也称作“数据库中的知识发现”或者

KDD）和虚拟化方法在日志数据上取得了巨大的成功，在本书的其他章节我们讨论了这些方法。它们使得组织能够找出日志数据中真正的异常，而不仅仅是“已知的问题”和“已知的正常情况”。

14.4　小结

我们冒昧地定义了一些日志记录领域的普遍真理。在规划和运营与日志相关的项目以及购买、构建和使用工具的时候记住它们很有用。

而且，避免常见的错误将使你的日志管理计划提高到新的水平，增强现有安全性和日志记录基础设施的价值。

参考文献

Marcus Ranum Logging Laws are from http://ranum.com/security/computer_security/archives/logging-notes.pdf (retrieved May 2012).

Chapter 15

第 15 章

日志分析和收集工具

本章要点

- 外包、构建或者购买
- 日志分析基本工具
- 日志信息集中化实用工具
- 日志分析工具——超越基本
- 商业化工具供应商

15.1 概述

日志管理和日志分析工具市场近年来有了显著的增长。许多工具集（如 grep 和 awk）已经集成到当今环境中使用的许多服务器和网络设备中。我们将研究开源解决方案和一些好的商业化工具，这些工具帮助你集中、管理和报告环境中的日志。这些工具包括基本的扁平文件搜索、集中化日志收集，以及加入了依从性相关报告和实时警报的更加健壮的工具。你适合何种工具取决于所要监控的系统数量以及组织的依从性需求。本章并不打算进行全面的说明，而是讨论当今广为使用的一些日志管理和分析工具，并且提供最适合于你的组织日常日志管理的工具和特性。

15.2 外包、构建或者购买

在许多公司中，安全和日志管理不是核心竞争力，是从事业务所需的成本而非收入来

源。多年来发生的一些引人注目的安全事故已经加强了安全性低下会付出代价的意识。在 2012 年最新的《 Verizon Breach Report 》中回顾了超过 855 次事故，结果表明，在这些攻击中有超过 1 亿 7 千 4 百万条记录遭到泄露（ Verizon 公司，2012 年）。这些事故使许多企业注意到，如果不选择正确的系统，或者没有足够的资源来监控系统，可能会付出很高昂的代价，也可能造成业务损失。

最近的许多事故（例如最近的 Global Payments 数据泄密事件）发生在需要遵循最严格的依从性原则（如支付卡行业数据安全标准 PCI DSS）的企业中。日志管理和审核是 PCI DSS 的关键部分。虽然我们不知道在这些事件中，合适的日志管理和日志分析系统起着什么样的作用，但是我们可以断定，快速地审核、关联和分析工具，在识别受影响的系统和客户，从而使公司能够弥补安全漏洞并持续运营中是至关重要的。

日志管理和日志分析系统应该接受和环境中任何关键 IT 系统一样的认真评估，才能做出购买、构建或者外包的决策。在许多情况下，购买和构建、购买和外包相结合，可能比提供单一方法更能够满足组织的需求。我们在本章将讨论各种不同的工具，你应该考虑购买、构建或者外包的利弊。

15.2.1　构建一个解决方案

由于开源日志管理解决方案的存在，以及对针对环境和需求定制和调整解决方案的渴望，许多组织选择构建日志管理解决方案。在许多情况下，这种方法使组织可以从现有的资源和有限的成本开始。与此相关，有一些优势和风险：

优势：

- 你能得到适用于环境的一个系统和解决方案（Chuvakin, 2008 ）。
- 你可以完成商业化或者开源解决方案中无法找到的功能，因为在许多情况下，你可以修改和更新系统的代码。
- 你可以选择和设计系统的平台、工具和过程。
- 获取系统的先期成本有限（甚至没有）。
- 这很有趣！

风险：

- 你拥有系统，必须为系统维护和持续满足依从性标准的任何更新分配资源和时间。
- 没有第三方支持。你就是支持人员！
- 如果构建该系统的关键人物离开公司，你能够雇用、保留和训练员工继续维护该系统吗？
- 系统是否能够处理来自组织各系统的日志，是否能够随着组织的成长而扩展？

15.2.2　购买

许多组织可能觉得，构建日志管理系统过于费时，如果系统开发不是公司的核心竞争

力，它们也可能没有资源投入到这些系统的构建和维护中。较大的组织还需要与供应商的支持协议，以保证正常运行时间和法律需求。基于开源产品自行开发的解决方案通常无法满足支持协议和法律需求。由于这些因素和其他一些因素，许多组织寻求购买日志管理系统。下面是购买日志管理系统时所应该考虑的。

优势：

- “付现自运”——付款之后，你就能获得一个日志管理和分析需求的解决方案。
- 订购的解决方案支持广泛的日志来源和格式。
- 支持协议通常包含正常运行时间和问题响应时间的服务协议。
- 产品更新和改进，包括满足依从性标准更改的更新。
- 可能购买服务和现场帮助，协助安装系统和训练内部人员。

风险：

- 除了初始系统成本之外，你现在将得到一个系统，需要雇用或者训练员工安装和使用它。你的组织应该考虑这对当前业务优先顺序的影响，以及初始系统成本之外的预算约束和持续的人员留存及教育成本。
- 你的组织有没有这样的员工，具备学习、使用和最大限度发挥所购系统的作用的技能？
- 系统中存在缺口，不能支持环境中安装的应用程序，或者与依从性需求相关的过程。
- 供应商成熟度、寿命和组织在未来更换供应商的风险。

15.2.3 外包

许多组织发现，在自身没有能力或者资源构建或者运营 / 维护所购买的解决方案时，外包是更好的替代方案。正如购买的选项一样，外包使得组织能够满足环境中运营的系统的正常运行时间、支持和法律需求。下面是外包时应该考虑的项目：

优势：

- 由别人去操心组织内的日志管理日常任务和依从性需求。这解放了你的资源，以便专注于其他核心业务（Chuvakin, 2008）。
- 外包限制了基础设施占用，外包供应商托管组织中安装的基础设施。
- 投入日志管理和审核日志日常活动及其他依从性需求的人员较少。

风险：

- 由别人操心你的问题，他们可能没有适合你的环境或者依从性需求的背景。
- 系统可能有缺口，不能支持环境中安装的应用程序，或者与依从性需求相关的过程。
- 组织失去了对其数据的控制。如果托管在组织外部，就会存在丢失数据的风险，在未来难以切换日志管理提供商。我们将在第 21 章中进一步讨论与云供应商相关的细节。
- 组织的数据量可能给供应商带来挑战，在向组织外传输数据时，可能影响你的系统 SLA。

- 访问日志数据可能受限于供应商提供的 API 和在线数据留存时间。

15.2.4　问题

研究本章的工具和解决方案时，考虑如下问题，以及每种工具如何帮助你回答这些问题。不管你的组织选择构建、购买或者外包日志管理和分析解决方案，这些问题都是切中要害的（Chuvakin, 2008）。

1）你是否收集和聚合了来自网络上所有数据源的全部日志数据？

2）日志是否安全地传输和保存？

3）打包的报告是否适合你的需求？你能够创建必要的报告，快速组织收集的日志数据吗？

4）你能在日志的任何内容上设置警报吗？

5）你是否每天查看日志数据，系统是否有能力聚合情报或者减少这些活动耗费的时间？你是否能证明自己有执行这项功能的工具，可以报告谁在何时审核了日志？

6）你是否能够对特定的数据执行快速、针对性的搜索？

7）你能够建立日志数据上下文，并在组织范围内进行关联，用于取证和其他运营任务吗？

8）你是否能够轻松地证明安全、变更管理和访问控制策略正在使用中且保持最新？

9）你是否能安全地与其他应用程序及用户共享日志数据？

15.3　日志分析基本工具

有一些简单而强大的工具内建在 Linux/Unix 操作系统中，可用于进行日志分析，它们可以通过互联网免费下载，或者作为许多组织已经订购的办公室生产力套件的一部分。我们在此将研究一些这类工具，并提供使用这些工具进行日志分析的例子。

15.3.1　grep

grep 是大部分 Linux 和 Unix 操作系统上都可以找到的命令行实用程序，能够根据模式或者正则表达式搜索输入文件（IEEE 和 The Open Group（国际标准开放组织），2004）。

这个实用程序已经移植到许多其他操作系统，包括 Windows，流行的移植版本有 GNU 的 grep for Windows（http://gnuwin32.sourceforge.net/packages/grep.htm），也是 Cygwin（http://www.cygwin.com/）的一部分。

本书的许多日志文件和示例说明，许多系统直接将日志记录为人类可读的文本文件。这些文件可以用 grep 轻松地搜索，寻找在日常日志审核和取证中有用的信息。grep 和本章介绍的许多搜索工具一样，严重依赖用户了解搜索项或者想要搜索的内容（参见表 15.1）。

表 15.1　日志审核中常见的搜索项目

在 Linux/Unix 上所要寻找的	
成功的用户登录	“Accepted password,”“Accepted publickey,”“session opened”
失败的用户登录	“authentication failure,”“new user,”“deleteuser”
用户注销	“session closed”
用户账户更改或者删除	“password changed,”“new user,”“delete user”
Sudo 操作	“sudo:…COMMAND=…,”“FAILED su”
服务故障（Chuvakin,2011）	“failed” or “failure”
在 Windows 上所要寻找的	
下面列出的事件 ID 用于 Windows 2000/XP。Vista/7 安全事件 ID 在这些事件 ID 上加上 4096。 下面的大部分事件在安全日志中，但是其中许多只在域控制器上记录	
用户登录 / 注销事件	成功登录 528,540；失败登录 529-537,539；注销 538,551
用户账户更改	创建 624；启用 626；更改 642；禁用 629；删除 630
密码更改	修改为原密码：628；修改为其他：627
服务启动或者停止	7035，7036
对象拒绝访问（如果启用审计）	560，567
在网络设备上所要寻找的	
对入站和出站活动都要注意 下面的例子选自 Cisco ASA 日志；其他设备的功能类似	
防火墙上允许的通信	“Built…connection,”“access-list…-permitted”
防火墙阻止的通信	“access-list…denied,”“deny inbound,”“Deny…by”
传输字节数（大文件？）	“Teardown TCP connection…duration…bytes…”
带宽和协议使用情况	“limit…exceeded,”“CPU utilization”
检测到的攻击活动	“attack from”
用户账户更改	“user added,”“user deleted,”“user priv level changed”
管理员访问	“AAA user…,”“User…locked out,”“login failed”
在 Web 服务器上所要寻找的	
对不存在的文件的过度访问 看上去像 URL 一部分的代码（SQL、HTML） 访问你没有实现的扩展 Web 服务停止 / 启动 / 失败消息 访问接受用户输入的“危险”页面 在不是你的文件上出现错误码 200	
用户认证失败	错误代码 401，403
无效请求	错误代码 400
内部服务器错误	错误代码 500

1. 使用 grep 进行日志分析

我们举一个 Apache 访问日志的例子。在组织内的许多 Web 服务器日志中，你会发现，

如果网站上保护的内容未经正确授权进行访问，就会出现 403 错误。我们用 grep 隔离这些错误，更仔细地看看发生了什么。

```
user$ grep -i -e "paybills.aspx?customer_acct=.* HTTP/1.1.* 403" /var/
   log/httpd/access_log
192.168.0.6 - - [10/Jul/2011:00:00:00 +0200] "GET /paybills.
   aspx?customer_acct=111111111 HTTP/1.1" 403 -
192.168.0.6 - - [10/Jul/2011:00:00:00 +0200] "GET /paybills.
   aspx?customer_acct=111111112 HTTP/1.1" 403 -
192.168.0.6 - - [10/Jul/2011:00:00:00 +0200] "GET /paybills.
   aspx?customer_acct=111111113 HTTP/1.1" 403 -
```

和第 9 章中的一些例子类似，我们使用一个正则表达式来隔离 paybills.aspx 页面中出现的 403 错误。现在看看结果，来自 192.168.0.6 地址的某个人递增客户账户号码，试图猜测系统中的有效账户，进行某些恶意活动。

2. grep 和正则表达式

grep 所有选项和正则表达式的更多细节，参见 GNU 网站：http://www.gnu.org/software/grep/manual/grep.html

15.3.2　awk

awk 是 Linux/Unix 平台中另一个强大工具，可以用作日志分析工具，从日志文件提取和隔离信息。

和 grep 一样，这个实用程序在其他操作系统上有许多移植版本。我们将主要关注 awk 的“print”命令，它能帮助我们从前一个例子中组合出恶意攻击者的所作所为。

awk 的 print 命令是一个强大的选项，允许你使用一个表达式，并显示该函数的输出。在我们的例子中，将从日志中捕捉特定的字段。awk 的 print 命令默认将空格作为一行文本中字段的分隔符，我们可以在字段表达式中使用“$[num]”引用特定字段。表 15.2 列出了日志文件中的字段和具体的数字字段标识符。

表 15.2　访问日志的 awk 字段标识符

awk 字段 ID	访问日志字段	示例数据
$1	IP 地址	192.168.0.6
$2	标识符[①]	—
$3	用户 ID	—
$4	条目日期和时间	[10/Jul/2011:00:00:00
$5	时区	+200]
$6	请求类型	“GET
$7	URL 和参数	/paybills.aspx?customer_acct=111111111

（续）

awk 字段 ID	访问日志字段	示例数据
$8	协议	HTTP/1.1”
$9	HTTP 状态码	403
$10	返回数据大小	—

①日志文件中的连字符表示该值不可用，或者不存在

注：表中的示例数据来自 Apache Web 服务器，使用常见的日志文件格式：

192.168.0.6--[10/Jul/2011:00:00:00+0200]“GET/paybills.aspx?customer_acct=111111111 HTTP/1.1”403

关于 awk 强大选项的更多内容，参见：

http://www.gnu.org/software/gawk/manual/gawk.html

现在，我们已经了解了更多关于如何解析日志文件字段的知识，让我们组合 grep 和 awk 的使用知识。假定我们只想看到 192.168.0.6 的攻击者访问的 URL，以及返回错误（如 403 代码）的页面和成功访问（状态码 200）的页面。在下面的例子中，我们将使用 grep，仅返回来自客户端 192.168.0.6 的日志条目，然后用 Unix 的“|”将信息传递给 awk 命令，以提取字段 7 的 URL 和字段 9 的状态码。

```
user$ grep '192.168.0.6' /var/log/httpd/access_log | awk '
   {print $9,$7}'
403 /paybills.aspx?customer_acct=111111111
403 /paybills.aspx?customer_acct=111111112
403 /paybills.aspx?customer_acct=111111113
200 /paybills.aspx?customer_acct=111111114
200 /change_password.aspx?customer_acct=111111114&pwd=1234
```

结合使用 grep 和 awk，我们得到 Web 服务器上可能发生的攻击和某个账户可能已经被侵入的清晰图景。从该例中可以看出，192.168.0.6 上的攻击者能够用暴力破解法猜测账户编号 111111114 是一个有效账户，并且更改了这个客户账户的密码！

15.3.3 Microsoft 日志解析器

本章中的许多工具是内建于 Linux/Unix 操作系统的工具，在 Windows 等其他操作系统上有一些移植版本。许多组织日常使用 Windows，Windows 也有自己的特殊日志和需求。

日志解析器（Log Parser）是来自 Microsoft 的免费工具。最新版本可以从 Microsoft 下载中心下载（http://www.microsoft.com/en-us/download/details.aspx?id=24659），正如 Microsoft 所称“日志解析器是强大的通用工具，提供了对基于文本（如日志文件、XML 文件和 CSV 文件）数据的通用查询访问，以及 Windows 操作系统关键数据源（如事件日志、注册表、文件系统和活动目录）的访问。”

日志解析器较为实用的功能之一是用类似 SQL 的语句查询 Windows 事件日志的能力。

下面的例子说明了使用工具生成 Windows 安全事件日志中事件 5159 有关报告的方法。

日志解析器工具支持许多不同的输出格式，在本例中，选择生成逗号分隔数值格式文件（CSV）。CSV 格式可以由多种不同应用程序加载。下面显示了 Microsoft Excel 中的日志解析器搜索结果，我们可以在 Excel 中过滤结果，进行日志分析。

图 15.1 展示了使用 Microsoft 日志解析器生成文本报告的一个示例。

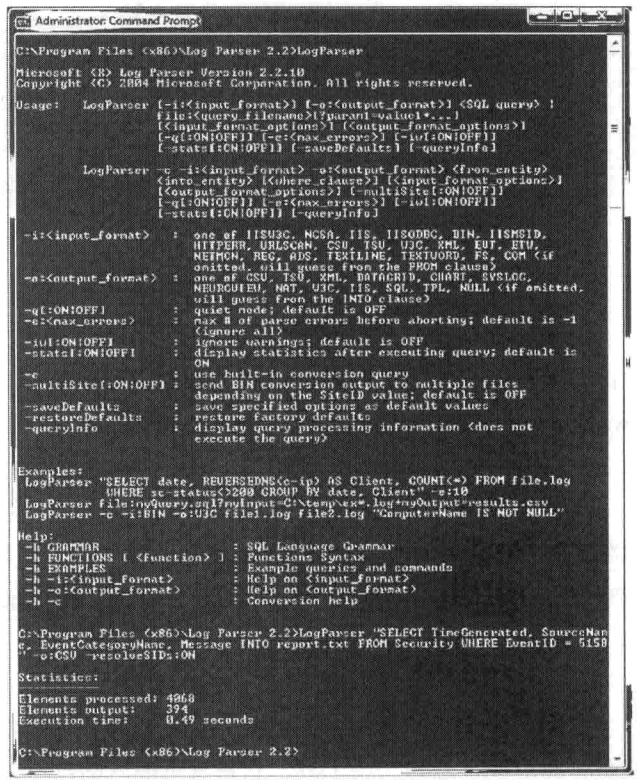


图 15.1　Microsoft 日志解析器

15.3.4　其他可以考虑的基本工具

我们已经介绍了进行人工日志分析的一些有力工具，并组合这些工具，让你在日志审核中进行更详细的分析。这类的工具还有很多，下面介绍几个可以添加到你的工具集中的工具：

1. tail

这个实用程序可以用于查看日志文件的结尾，或者在日志最后发生的当前操作（http://www.gnu.org/software/coreutils/manual/coreutils.html）。

图 15.2 展示了 Microsoft Excel 中的搜索结果。

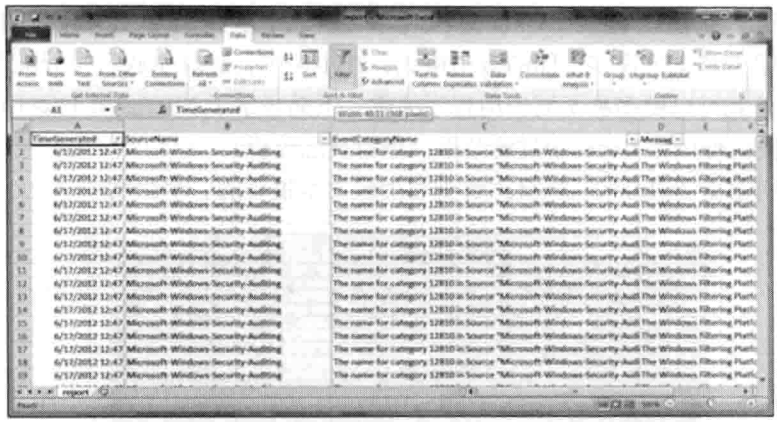


图 15.2　Microsoft Excel 中的搜索结果

2. head

这个实用工具是 tail 的反面，可以用于检索大日志文件的开头部分，使其更加容易用其他工具查看（http://www.gnu.org/software/coreutils/manual/coreutils.html）。

3. Sed

Sed 是类似 awk 的解析工具，在文本搜索和替换中很有用，可以使日志更加清晰易读，也可以用于格式化日志输出，使其更加容易被其他工具使用（http://www.gnu.org/software/sed/manual/sed.html）。

4. Logwatch

Logwatch 是离线解析和审核日志的实用工具，它具有可插入的接口，允许你根据需求定制。该工具可以从 http://sourceforge.net/projects/logwatch/ 下载。

5. Lire

Lire 是一个应用程序套件，可以用来从日志文件生成自定义报告。在这个方面它与 Microsoft 日志分析器（Log Analyzer）类似，你可以从日志文件中过滤和生成一个报告。Lire 支持来自大部分非 Windows 设施和软件的各种日志文件。这个工具是 Microsoft Log Analyzer 在非 Windows 系统上的一个很好补充。更多信息参见该工具的网站：http://www.logreport.org/lire.html。

15.3.5　基本工具在日志分析中的作用

在前几个小节中，我们展示了使用基本工具的一些很好的方法，这些工具是免费的，或

者是你的环境中已经采用和管理的系统和软件包的一部分。示例演示了如何使用这些工具搜索和提炼日志数据，以隔离攻击。这些工具最适合用于小型组织和人工日志分析及审核你应该考虑到下面列出的这些工具的一些局限性，以及它们如何融入一个环境、组织的依从性需求，和必需的时间和资源需求。在许多情况下，随着日志尺寸的增长和环境的发展，这些工具可能只适合于取证。

需要考虑的因素有：

- 你需要知道所要搜索的内容。在例中，我们知道想要搜索的文本，然后使用 grep 和 awk 及其他工具，根据非常明确的文本对数据进行切片。如果你搜索趋势或者统计异常，这些工具可能并不适合。
- 在大文件上运行的时间很长。如果你的组织有数以 TB 计的日志信息，找到特定字符串所需要的时间长得难以想象。这些工具通常会搜索整个文件，以找出你想要的内容。
- 在不同日志之间关联的能力有限。如果你的日志文件分散在多个设备和多个日志文件中，同步防火墙、Web 服务器和客户端工作站中的不同日志很难。这就很难弄清整个网络路径中攻击是如何发生的。

15.4　用于集中化日志分析的实用工具

在过去的几章中，我们讨论了在组织中集中日志记录信息的必要性，使你能够关联整个环境中的日志，从而方便实现 PCI DSS、HIPAA 和 SOX 等收集与分析依从性需求。有一些免费的工具可以帮助你在环境中集中日志记录信息。我们将关注 syslog、syslog-ng、rsyslog 和 Snare 等工具，以及它们帮助集中化日志的方式。

15.4.1　syslog

syslog 是目前最为普遍的跨平台日志记录解决方案。网络设备、Unix/Linux 系统上都有内建的 syslog 日志记录支持，还有许多编程语言库扩展，可以方便应用程序开发人员在应用程序中添加 syslog 日志记录。

syslog 有两个主要的组成部分。syslog 客户端存在于系统之上，生成日志，可以配置为将其日志发送到本地或者集中化 syslog 服务器。最后，有一个 syslog 守护进程（或者服务器端进程）从配置为从发送数据的其他 syslog 客户端接收日志。

syslog 在本书中我们讨论的集中化日志信息关键领域中都很强大，但是 syslog 中缺少的如下功能可能正是你的环境中必需的。

1. 有保证的消息传送

在 syslog 的常见实现中，syslog 通过 UDP 端口 514 对日志消息进行通信和中继。在繁

忙的网络上，用户数据报协议（UDP）流量可能会丢弃消息，以便更高优先级的流量（最明显的是传输控制协议（TCP）流量）通过。所以，从主机发出的每条日志消息都不能保证记录到集中化 syslog 服务器上。

更多细节请参见 RFC-5426（http://tools.ietf.org/html/rfc5426）中的“Unreliable Delivery”。

图 15.3 展示了集中化 syslog 日志的概要描述。

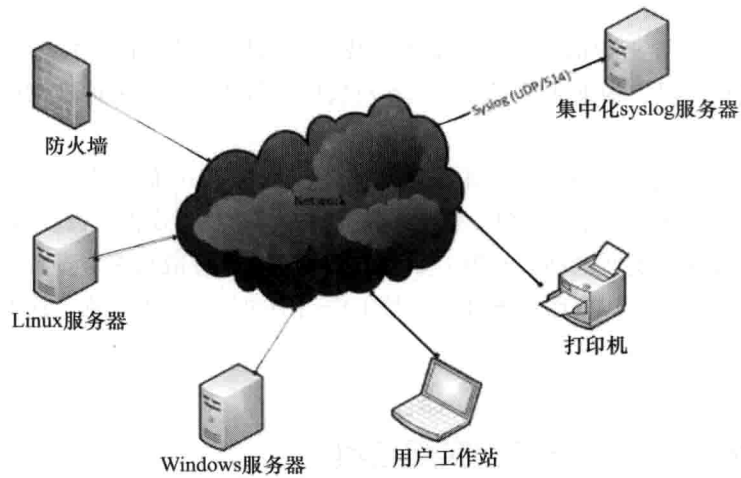


图 15.3　集中化 syslog 日志的概要描述

2. 日志消息的安全传输

尽管在 RFC-5425 中提出（http://tools. ietf. org/html/rfc 5425），但是 syslog 实现中很少为了避免在网络上被窃听和篡改而加密日志消息。这对于本书提出的许多日志记录原则来说都是很重要的，对于组织来说，随着它们的远程站点越来越多，并且采用云中的虚拟化主机，这一点也越来越重要。

3. 日志源信息维护

在某些 syslog 环境中，使用一个中继服务器从远程站点或者网段收集日志，并将日志转发到集中 syslog 服务器。在 syslog 的许多常见实现中，日志记录原始生成来源的信息在中继过程中丢失。这使日志消息的来源确定几乎不可能实现。

15.4.2　Rsyslog

Rsyslog 是另一种开源选择，在许多 Linux 发布版本中可以免费使用。syslog-ng（见第 5 章）的功能有一些差异，但是整体来说，两者在功能集上基本兼容。和 syslog-ng 相比，选择 rsyslog 有如下关键好处：

- Rsyslog 是 Redhat 和 Fedora Linux 系统的默认日志记录器，对于安装了大量 Redhat 的组织是更为自然的选择。
- Rsyslog 最近增加了对 Hadoop（HDFS）的支持。为了增强日志挖掘的安全能力而迁移到 Hadoop 的组织将能够用这一功能集成 rsyslog，只需要编写有限的自定义脚本，将数据加载到 Hadoop。

15.4.3　Snare

环境中的 Windows 系统在 Windows 事件日志中记录大量有价值的信息。遗憾的是，事件日志是 Windows 专利技术，对 syslog 风格的消息没有原生支持。syslog-ng 和 rsyslog 有付费的选项，可以从 Windows 系统读取事件日志信息。但是，Snare 是另一个免费的替代方案。

Snare（http://www.intersectalliance.com/projects/index.html）Windows 代理可以通过 syslog 发送 Windows 事件。Snare 有代理和服务器收集选项。但是，免费的 Snare 服务器选项被限制在日志收集上。如果你的预算有限，还有一些免费的方案能够提供收集和分析，我们将在本章后面介绍。

图 15.4 展示了 Snare 网络配置屏幕。

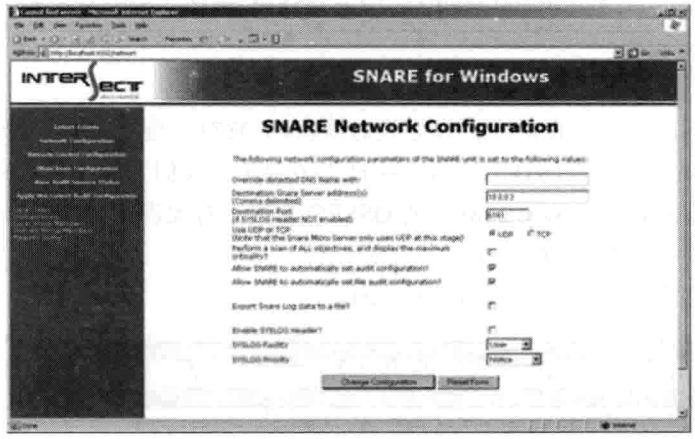


图 15.4　Snare 网络配置屏幕

要采用 Snare Windows 代理通过 syslog 发送事件日志，你必须在代理安装后进行一些小的调整。安装之后，进入网络配置，并进行如下修改：

- 将目标端口设置为默认的 syslog 端口 514。
- 启用 Syslog 报头。

15.5　日志分析专业工具

我们已经介绍了日志分析的基本工具。本节介绍进行日志信息深入分析的工具。在这里

介绍的某些工具允许你实时生成事件警报，生成帮助你实现依从性的摘要报告，一般还会为你提供某些功能，减少日常日志审核和在事件发生时的日志取证所需的时间。

15.5.1 OSSEC

OSSEC（www.ossec.net）是一个出色的开源日志留存和分析工具。这个工具集包含许多平台的代理，支持从现有 syslog 接收日志，包括对无代理选项的支持，该选项能够检查许多无法安装 OSSEC 代理的平台上的文件完整性，甚至可以安装在 VMware 主机系统上，对虚拟化基础设施进行监控。该工具中许多预装的规则能够帮助你开始实时警报，还有一个基于 Web 的用户界面，可用于进一步的警报和日志审核，大大减少了日志分析的人工过程。OSSEC 是一个非常轻量级的进程，集中化管理服务器只有 Unix/Linux 选项。本节我们将重点介绍 OSSEC 中与日志分析相关的许多功能。对于想要管理和安装 OSSEC 工具集的读者，有许多出色的网站和书籍。

OSSEC 有一个活跃的用户和开发人员社区并经常更新，提供新功能和缺陷修复。2009 年，Trend Micro 收购了 OSSEC，继续保持该工具的开放和免费。这种关系对于已经与 Trend Micro 建立联系的组织有利，Trend Micro 已经为 OSSEC 提供了商业化支持。

1. 留存所有日志信息的 OSSEC 设置

为了遵循组织的日志留存策略，你应该对默认的 OSSEC 配置进行一些调整。默认情况下，OSSEC 只记录生成警报的日志条目。在日志留存中，我们一般希望留存所有日志信息。我们可以更新 /var/ossec/etc/ossec.conf 中的 OSSEC 配置，对此进行调整。简单地将“logall”选项设置为 yes 就可以留存所有日志。

图 15.5 展示了留存所有日志的 OSSEC 设置。

图 15.5　留存所有日志的 OSSEC 设置

默认情况下，OSSEC 将使用系统的 /var 分区中的目录留存日志。你应该分配系统磁盘空间的一大部分给这个分区，或者更改默认日志位置，确保有足够的空间存放日志。

2. OSSEC 实时警报

OSSEC 中的日志分析有力工具之一是内建的规则和编写自定义规则的能力。规则为你提供根据接收到的日志事件生成实时警报的能力。默认情况下，所有规则保存在服务器上的 rules 文件夹中。OSSEC 自带许多内建规则，以根据来自防火墙、Windows、Linux/Unix、McAfee 和 VMware 等应用程序的事件，以及常见的攻击和 rootkit 生成警报。内建规则的文档可在 http://www.ossec.net/doc/rules/rules/index.html 上找到。

规则语法用 XML 格式编写。规则可以根据单一事件、一条原子（简单）规则或者事件发现次数、复合（关联）规则生成。下面展示内建原子规则的一个例子：

```
<rule id="18100" level="0">
   <category>windows</category>
   <description>Group of windows rules.</description>
</rule>
…
<rule id="18105" level="4">
   <if_sid>18100</if_sid>
   <status>^AUDIT_FAILURE|^failure</status>
   <description>Windows audit failure event.</description>
</rule>
```

在上述规则中，简单地搜索事件中的“AUDIT_FAILURE”或者“failure”消息。我们可以用复合规则进一步调整，以便关注更需要立即引起重视的事件。

```
<var name="MS_FREQ">6</var>
…
<rule id="18153" level="10" frequency="$MS_FREQ" timeframe="240">
   <if_matched_sid>18105</if_matched_sid>
   <description>Multiple Windows audit failure events.</description>
</rule>
```

在上述的内建复合规则中，我们用故障原子规则 18105 创建这条复合规则，如果在 4 分钟（240 秒）中发生 6 次故障，则触发该规则。

我们还可以构建自己的规则，这些规则可能更切合组织的实际。对于你的组织，你可能想要在下午 5 点至上午 9 点的非业务时间中发生 10 次登录失败之后发出警报，此时可以编写如下复合规则：

```
<rule id="100999" level="10" frequency="10" timeframe="300">
   <if_matched_sid>18105</if_matched_sid>
```

```
    <time>5 pm - 9 am</time>
    <description>Windows Logon Failure occurred 10 times after business
    hours</description>
</rule>
```

规则引擎可以帮助你的团队满足许多依从性框架的日志审核需求。原子和复合规则的结合还将帮助你减少使用我们讨论过的基本工具集进行人工审核所需的时间，你还可以调整警报，减少假阳性。

3. OSSEC 日志留存策略和存储选项

默认情况下，OSSEC 将把日志信息存储在 /var/ossec/logs/alerts。OSSEC 在基于日志事件生成的年份和月份的目录结构中保存日志。下面是 OSSEC 安装中的典型日志留存结构：

```
/var/ossec/logs/alerts/2011
/var/ossec/logs/alerts/2011/Jan
/var/ossec/logs/alerts/2011/Feb
/var/ossec/logs/alerts/2011/Mar
...
```

这种目录结构使得我们很容易编写一个脚本，清理超过留存期的旧日志。简单命令“rm–f/var/ossec/logs/alerts/2011/Jan”就能删除从 2011 年 1 月至今的所有日志。

许多组织可能希望编写自己的自定义报告，或者扩展 OSSEC 的搜索和取证功能。如果你没有程序员或者不想修改和维护自定义的 OSSEC 源代码，另一个选项就是将 OSSEC 警报发到一个数据库，这样就可以编写 SQL 查询，获得所需的报告或者信息。OSSEC 目前支持使用 MySQL 和 PostgreSQL 数据库保存警报。将警报发送给数据库的更多信息可以在 http://www.ossec.net/doc/manual/output/database-output.html 找到。

15.5.2 OSSIM

OSSIM 不仅能进行日志管理和留存，它应该归入安全信息和事件管理（SIEM）系统。OSSIM 提供了所谓的“5 个必不可少的安全功能”——资产发现、漏洞评估、威胁检测、行为监控和安全智能。它在内部使用了如下的开源工具，提供这些功能。

- Arpwatch 用于 MAC 地址异常检测。
- P0f 用于被动 OS 检测和 OS 变更分析。
- Pads 用于服务异常检测。
- Nessus 用于漏洞评估和交叉关联（入侵检测系统（IDS）和漏洞扫描器）。
- Snort 用作入侵检测系统（IDS），也用于和 Nessus 进行交叉关联。
- Tcptrack 用于会话数据信息，可以收集攻击关联所需的有用信息。
- Ntop 构建一个令人赞叹的网络信息数据库，用于异常行为检测。

- Negios 用于根据主机资产数据库监控主机和服务可用性信息。
- Osiris 是基于主机的入侵检测系统（HIDS）。
- Snare 是 Windows 系统的日志收集器。
- OSSEC 是一个基于主机的 IDS。
- OSSIM 还包含了自己的工具，包括一个通用关联引擎，具备逻辑指令支持和使用插件进行日志集成的功能。

你将会注意到，OSSIM 包含了我们已经讨论过的两个工具：OSSEC 和 Snare。OSSIM 包含了一组引人注目的图标和数据视图，它们尚处于初期阶段，或者不是 OSSEC 的一部分。AlienVault 提供了免费和商业化的版本。如果组织寻求一种完整的安全信息管理系统而不仅仅是日志管理和留存系统，应该考虑评估 OSSIM。更多的信息参见 http://www.alienvault.com/。

15.5.3　其他值得考虑的分析工具

本节我们已经介绍了许多出色的工具，当然还有许多其他工具。下面是一些可以考虑添加到你的工具箱的优秀软件。

1. Logsurfer

和本节提到的许多工具不同，Logsurfer 是一种实用的实时日志审核工具。它的另一个好处是尝试将可能被看作事件的消息组合成单一逻辑事件。例如，考虑下面的 FTP 事件序列。

```
somehost tcpd-ftpd[14311]: connect from host.some.where
somehost ftpd[14311]: connection from *@host.some.where [42.42.42.42]
somehost ftpd[14311]: USER anonymous
somehost ftpd[14311]: PASS password
somehost ftpd[14311]: failed login from host.some.where [42.42.42.42],
   anonynous
somehost ftpd[14311]: SYST
somehost ftpd[14311]: PORT
somehost ftpd[14311]: cmd failure - not logged in
somehost ftpd[14311]: LIST
somehost ftpd[14311]: QUIT
somehost ftpd[14311]: FTP session closed
```

只看到其中一条消息，可能造成假阳性安全事件。但是，如果我们将这些消息的内容当成一个整体看，就会发现某人试图匿名登录，然后注销。对这些日志消息的分析将用如下内容代替：

```
somehost ftpd[14311]: cmd failure - not logged in
```

这个工具可以从 http://www.crypt.gen.nz/logsurfer/ 下载。

2. LogHound

LogHound 更像一个研究工具，它试图用广度优先算法挖掘日志文件，找出频繁出现的行模式。这个工具的更多细节和研究论文可以在该工具的网站上找到：http://ristov.users.sourceforge.net/loghound/。

3. Log2Timeline

Log2Timeline 在取证分析中很有用。该工具从非传统来源（如回收站、Firefox 历史、代理服务器数据等）提取时间轴信息。这在怀疑数据或者日志已经被修改或者更新时可能有用。该工具可以在 http://log2timeline.net/ 取得。

15.6 商业化日志工具

安全业界领先企业有许多杰出的日志分析和留存解决方案，涵盖了从可以自行购买运行的解决方案到允许你将日常日志管理外包给第三方提供商的日志管理服务的范围。我们将在这里评估一些产品，但是这绝对无法包含所有可用的解决方案。Gartner（www.gartner.com）和 Security Scoreboard（http://www. securityscoreboard.com/）有可供进一步研究的资源。

免责声明

本书对本节中提到的商业化产品不做明确的提倡或者推荐。和环境中任何购买行为一样，我们建议对任何商业化产品进行完整的评估，确定它是否符合你的组织需求。和以往一样，提出许多问题，尝试和供应商一起进行概念验证，然后购买适合你的解决方案。

15.6.1 Splunk

Splunk（www.splunk.com）是一个商业化产品，提供许多我们在介绍 OSSEC 时提到的出色功能。Splunk 允许你集中日志，用于取证和关联，也能根据需要进一步调查或者采取行动的事件类型，生成实时警报。Splunk 与众不同的地方是它支持广泛的日志源、审核活动的实时仪表盘、可自定义报告和仪表盘以及有助于将 Splunk 集成到安全基础设施的供应商支持 API。

1. 免费版与企业版的对比

Splunk 的好处之一是它有免费和企业版本。你可以在 Splunk 网站上建立一个免费账户，并下载软件开始使用。但是，免费版本有如下关键限制。

- 免费版本每天最多处理 500MB 日志。虽然这听上去很多，但是大中型企业很快就会

达到这个限值，几个真正繁忙的 Web 服务器就可能立刻超过这一限制。

- 实时监控和警报只在 Splunk 企业版中可用，但是，你可以试用企业版 60 天来评估这一选项。
- 基于角色的身份认证只在企业版中可用。

2. 日志源支持

Splunk 支持广泛的数据输入。它支持我们在本节中已经讨论过的许多输入源，如标准 syslog、通过 TCP 的 syslog，当然还有使用 TLS/SSL 加密的 syslog。在此基础上，Splunk 具备“转发器”，可以从文件和目录读取信息，读取 Windows 事件和注册表信息，并支持读取许多 Web 服务器日志。Spluk 有丰富的“处方”，可以读取 Web 日志、J2EE 日志、Windows 性能指标等。对于较不常见的输入，Splunk 可以定义输入的细节，作为它处理和索引数据的“处方”。

3. 仪表盘和视图

Splunk 允许用户以仪表盘或者视图的形式，创建自己的数据视图。这些视图完全可以自定义，和我们讨论过的工具相比，这些功能对于某些组织来说是关键的差异。有些组织有许多自定义报告需求，或者需要为组织中不同利益相关方提供不同数据视图，它们会发现，Splunk 的这种特性显著地减少了从前面介绍的手工工具集中获得报告的时间。

15.6.2　NetIQ Sentinel

NetIQ Sentinel（https://www.netiq.com/products/sentinel/）和前面讨论的 OSSIM 类似，不仅是日志管理和分析产品，而应该归入 SIEM 的类别。Sentinel 包含了异常检测和身份管理信息，作为处理事故响应和事件取证的额外来源。如果你需要的不仅是日志管理，NetIQ Sentinel 的企业版本可以为你的组织提供完整的安全信息管理工具。

这个产品于 2010 年从 Novell 收购而来，免费版被限制为仅有日志管理功能，最大处理速度限制在每秒 25 个事件。可以在 http://www.novell.com/promo/slm/slm25.html 创建一个账户。超过以上限制的负载要求你购买 Sentinel 产品。该产品具有强大的搜索和报告功能。

15.6.3　IBM q1Labs

QRadar（http://q1labs.com/products.aspx）是来自 IBM 的日志管理解决方案。IBM 于 2011 年收购了 q1Labs 及其产品线。该产品本来有一个免费版本，但是从收购之后就不再可用。和我们讨论过的许多产品一样，QRadar 有广泛的日志源支持，以及广泛的搜索和报告功能。和 Qradar 一起被收购的还有其他产品线，包括一个为需要完整安全信息管理工具的企业提供的 SIEM。QRader 与众不同的特点之一是它可以为许多当前依从性框架对报告进行调整，

支持如下审计：

- 支付卡行业数据安全标准（PCI DSS）
- 北美电力可靠性公司（NERC）
- 健康保险隐私及责任法案（HIPAA）
- Sarbanes-Oxley（SOX）和联邦信息安全管理法案（FISMA）

15.6.4 Loggly

Loggly（http://loggly.com/）是一个云日志提供商，你的日志数据将被存储在云中。我们将在第 21 章中更详细地介绍它，以及在云中保存数据时需要考虑的因素。该产品支持我们已经介绍的许多数据源，以及通过 REST API 编程发布数据，通过 HTTP 和 HTTPS 传递数据。这个产品相对新颖，你应该评估其分析引擎的报告能力，确定它是否满足你的依从性需求。Loggly 的关键特征是为留存日志而在环境中安装的附加硬件有限，需要由其他人维护和升级你的日志系统。你可以研究第 21 章和该公司的网站，了解这个解决方案是否适合你的组织。第 21 章中还有使用 Loggly 的 Web 界面发送和读取日志的示例。

15.7 小结

本章介绍了进行日志收集和分析的一些最常用工具，包括开源和商业化系统。而且，还提供了对于如何选择工具的一般指导原则。

参考文献

AlienVault. OSSIM the open source SIEM. <http://www.alienvault.com/>.

Chuvakin, A. (2008). How would you do it? Selecting a log management approach. <http://www.slideshare.net/anton_chuvakin/choosing-your-log-management-approach-buy-build-or-out-source>.

Chuvakin, A. (2011). Official page for critical log review checklist. <<http://chuvakin.blogspot.com/2010/03/simple-log-review-checklist-released.html >.

GNU.org. GNU Grep 2.13. <http://www.gnu.org/software/grep/manual/grep.html>.

Ossec.net. Rules/decoders documentation. <http://www.ossec.net/doc/rules/rules/index.html>.

Rsyslog.com. Rsyslog home. <http://www.rsyslog.com/>.

Syslog.org. Syslog.org home. <http://www.syslog.org/>.

The IEEE and The Open Group (2004). The open group base specifications issue 6. <http://pubs.opengroup.org/onlinepubs/009695399/utilities/grep.html>.

Verizon Corporation (2012). 2012 Data breach investigations report. Retrieved July 8, 2012, from Verizon Business: <http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2012_en_xg.pdf>.

第 16 章 Chapter 16

日志管理规程

本章要点

- 假设、需求和预防措施
- 常见角色和职责
- PCI 和日志数据
- 日志记录策略
- 审核、响应和升级规程以及日志审核的工作流验证
- 日志簿——调查异常证据
- PCI 依从性证据包
- 管理报告
- 定期运营任务附加信息

16.1 概述

现在你已经熟悉了日志管理、日志分析、关联、过滤等主题，下一步是什么呢？日志记录和管理的目标本质上是为你提供环境中的态势感知（SA），这样你就可以在网络中发生某些情况时进行评估、响应，并在必要时升级。SA 部分通过启用日志数据收集、分析和留存来实现。这些主题在本书中已经作了介绍，本章将介绍日志数据审核、响应和升级。

你的组织很可能因为如下的某些原因而实施日志记录：

1）因为你想要保护公司的资产（知识产权、财务数据等）。

警告 不保证 PCI 依从性

作者不能保证实施本章中的规程和工作流程就能达到 PCI 依从性或者通过 PCI 验证。本章中的素材只用于教育的目的。

2）你所处行业（银行、医疗、信用卡处理等）要求监管依从性，确保你能够对抗外部和内部威胁、数据丢失等。

3）你就是希望记录所有日志。

现在，第 2 种原因令人感兴趣。许多法规不仅从过程和规程的角度规定你必须做的事情，还要求你能够证明自己真的遵循和维护你的策略和规程。这可能意味着从提供最新的网络系统访问权限文档，到制作备份文档的报告（例如登录和注销记录）的一切事情。

本章中使用的方法是介绍可重复和可扩展的工作流，为你提供深入的领悟和理解，使你能够按照需要实施这些过程。为了重申和解释这些要点，我们将提供支付卡行业（PCI）数据安全标准（DSS）的例子。

从现在开始，本章的余下部分将使用 PCI DSS 的概念。

16.2 假设、需求和预防措施

这些关键项目对于日志记录、日志管理和日志审核的成功必不可少。在日志审核、响应和升级规程投入使用之前，假定如下需求已经得到满足。

注释 PCI DSS 简介

PCI DSS 是一组技术和运营需求，意在保护信用卡持卡人数据免遭违规使用。商户、处理设备、发卡行和服务提供商在 PCI 中都有利害关系。该标准的终极目标是在全球采用这些需求。PCI 的全部内容超出了本章的范围（《Requirements and Security Assessment Procedures》，2010）。

16.2.1 需求

必须有一组需求，本章描述的运营规程才能有效地使用：

1）创建日志记录策略（参见本章的 16.5 节），以编集 PCI DSS 日志相关需求，以及其他监管和运营日志需求。

2）在范围内的系统上启用日志记录。

3）日志记录的中断和终止本身必须记入日志、接受监控。

4）记录 PCI DSS 文档规定的事件。

5）生成满足 PCI DSS 日志记录需求的日志。

6）范围内的系统时间和可靠的时间服务器（NTP 等）同步。

7）所有日志记录系统的时区已知并作记录，可以和日志一起审核。

第 2 ~ 7 条可以适用于几乎每个环境，而第 1 条看起来相当特殊。但是应该注意的是，第 1 条（日志记录策略）是最佳实践，无论如何应该遵循。

16.2.2　预防措施

需要采取如下的预防措施，使日志可以用于 PCI DSS 依从性、其他法规和安全、取证及运营需求：

- 关键预防措施：在特定系统上记录其操作的个人不能作为负责该系统日志审核的唯一当事人。
- 关键预防措施：PCI DSS 强制实施日志安全措施，对日志的所有访问应该记录和监控，识别终止或者影响日志存在和质量的企图。

这些预防措施的主要思路是确保系统完整性。本质上，没有一个人应该拥有能够掩盖自己或者其他人踪迹的控制权。

16.3　常见角色和职责

表 16.1 总结了本章中提到的涉及审核、升级和响应过程的常见角色。

表 16.1　常见角色和职责总结

角　色	职　责	涉及工作示例
应用程序管理员	管理应用程序	配置应用程序日志记录设置，可能进行日常日志审核
系统或者网络管理员	管理底层操作系统或者网络	配置应用程序日志记录设置，可能进行日常日志审核
应用程序业务负责人	负责应用程序的业务经理	批准日志记录和日志审核所需的应用程序配置更改
安全管理员	管理一个或者多个系统 / 应用程序的安全控制	配置安全和日志记录设置，进行日常日志审核
安全分析人员	处理运营安全性过程	访问安全系统并分析日志和其他数据
安全主管或者经理	监督安全策略、过程和运营	负责日志审核规程、更新规程
事故响应人员	参与安全事故响应	处理安全事故，在响应过程中审核日志

本章将介绍这些角色和职责，应该指出的是，你的组织可能没有定义所有角色，这可能是因为你的组织很小，一人兼任多种角色。

身兼多职可能导致利益冲突问题

在日志数据审核、响应和升级中，同一个人兼任多个角色往往不是好主意。可将其视为利益冲突，应该尽可能避免。

16.4 PCI 和日志数据

本节介绍 PCI DSS 日志记录基本知识及其要求。《Requirements and Security Assessment Procedures》（需求和安全评估规程，2010）提供了 PCI DSS 规章的完整细节，包括分段、需求和本章引用的其他项目。

现在，日志记录和监控不限于需求 10，实际上遍及所有 12 条 PCI DSS 需求；PCI DSS 对于日志记录和监控的强制关键领域是需求 10 和需求 11 的各个部分。

注释 **需求 10-12**

顾名思义，为了达到依从性，你必须实现需求；它是必要的条件。在 PCI DSS《Requirements and Security Assessment Procedures》（2010）文档中，需求 10 陈述如下："跟踪和监控对网络资源和持卡人数据的所有访问"。

需求 11 陈述如下："定期测试安全系统和过程。"需求 12 陈述如下："维护一个处理所有个人信息安全的策略"。

16.4.1 关键需求 10

我们将逐行介绍关键需求，然后深入细节、示例和实现指南。

10. 1

具体地说，需求 10.1 涵盖的是"建立一个过程，将对系统组件的所有访问（特别是使用根用户等管理权限完成的访问）与每个单独用户联系起来。"（《Requirements and Security Assessment Procedures》，2010）。这确实是一个有趣的需求；它不仅强制日志的存在或者建立日志记录过程，还提到日志必须与个人绑定（不是计算机或者生成日志的"设备"）。这个需求常常给 PCI 实施者造成问题，因为许多人认为日志是"人的操作记录"，而现实中他们只拥有"计算机的操作记录"。将后者映射到实际用户往往带来额外的挑战。顺便说一句，PCI DSS 需求 8.1 强制组织"在允许用户访问系统组件或者持卡人数据之前，要为他们分配唯一的 ID"（《Requirements and Security Assessment Procedures》，2010），这一条目使日志在此更加有用。

10. 2

接下来，10.2 小节定义了需要记录的系统事件（或者，使"事件可以重建"）的最小列表。这些需求的动机是为了评估和监控用户操作，以及可能影响信用卡数据的其他事件（如系统故障）。

下面是 PCI DSS（v.2.0；《Requirements and Security Assessment Procedures》，2010）需

求（必须记录的事件）中的列表：

10.2.1　访问持卡人数据的所有单独用户。

10.2.2　具有根或者管理权限的任何个人进行的所有操作。

10.2.3　对所有审计跟踪的访问。

10.2.4　无效的逻辑访问企图。

10.2.5　标识和身份认证机制的使用。

10.2.6　审计日志初始化。

10.2.7　系统级对象的创建和删除。

可以看到，上述列表覆盖了数据访问、特权用户操作、日志访问和初始化、失败和无效的访问请求、身份认证和授权决策和系统对象更改。重要的是，这样的列表来源于 IT 治理“最佳实践”，这些最佳实践指出，要监控访问、身份认证、授权变更管理、系统可用性和可疑活动。

10.3

而且，PCI DSS 需求 10 更加深入，包括了每个事件中需要记录的具体数据字段或者数值。它们提供了健康状况的最低需求，各种 IT 平台上的日志记录机制一般都能超过这个要求。

这些字段是：

10.3.1　用户标识。

10.3.2　事件类型。

10.3.3　日期和时间。

10.3.4　成功或者失败的指示。

10.3.5　事件来源。

10.3.6　受影响的数据、系统组件或者资源的标识或者名称。

可以看到，上述最小列表包含了事故分析所需的基本属性，能够回答如下问题：何时、谁、何地、发生何事、来自何处。例如，如果试图发现谁修改一个信用卡数据库，将所有事务的详情复制到一个隐藏文件（典型的内部人员权限违规使用），了解上述所有记录非常有用。

10.4

下一个需求 10.4，处理常常被忽略但是很关键的一个需求：所有日志具备准确而一致时间的需求（William 和 Chuvakin，2012）。时间和安全事件监控的紧密结合似乎相当简单。在家庭或者小型办公室网络中，系统时间往往是随机的。不管你的服务器时间设成什么，为了设计有一定可靠性的网络，你的系统应该配置为从可靠来源（如网络时间协议（NTP）服务器）获得时间同步。

10.5

接下来，人们必须解决日志的机密性、完整性和可用性（CIA）。PCI DSS 的 10.5.1 小

节介绍了机密性："将对审计跟踪的查看限制在与工作相关的需求上"（《Requirements and Security Assessment Procedures》，2010）。这意味着，只有必须查看日志才能完成工作的人才能够查看日志，原因之一是和身份认证相关的日志总是包含用户名。虽然不是真正的秘密，但是用户名称信息提供了密码猜测所需的50%信息（另外50%是密码）。而且，因为用户错误输入证书，而在日志中显示密码的情况并不少见。编写质量低劣的Web应用程序可能造成密码和Web统一资源定位符（URL）一起记录在Web服务器日志中。

接下来是"完整性"。根据PCI DSS的10.5.2小节，必须"保护审计跟踪文件免遭未授权修改"（《Requirements and Security Assessment Procedures》，2010）。这是显而易见的，因为如果日志被未授权方（或者任何人）修改，它们就不再是系统和用户活动的客观评估跟踪。

但是，人们不仅必须保留来自恶意用户的日志，还应该保留系统故障和系统配置错误结果的日志。这涉及日志数据的"可用性"和"完整性"。PCI DSS 10.5.3明确指出，必须"立即将审计跟踪文件备份到集中日志服务器或者难以修改的媒介"（《Requirements and Security Assessment Procedures》，2010）。确实，将日志集中到一个或者一组用于日志分析的服务器，对于日志保护和增加日志实用性都是不可或缺的。将日志备份到CD或者DVD（或者磁带）是这一需求的另一种结果。人们应该始终记住，磁带上的日志不容易访问，在出现事故时也无法搜索。

许多网络基础设施（如路由器和交换机）被设计为将日志放到外部服务器，在设备自身上只保留极少量（或者没有）日志。因此，对于这些系统来说，集中化的日志最为重要。PCI DSS的需求10.5.4说明了这种需求："将无线网络的日志复制到内部LAN上的日志服务器中。"

为了进一步降低日志修改的风险，并证明这样的修改没有发生，需求10.5.5要求"在日志上使用文件完整性监控和变更检测软件，确保现存日志数据不能在没有警告的情况下更改"（《Requirements and Security Assessment Procedures》，2010）。与此同时，在日志文件中添加新的日志数据不应该生成警报，因为日志文件不断增长，不会自己压缩（除非日志轮转或者存档到外部存储）。文件完整性监控系统使用加密哈希算法，将文件和一个已知的正确副本比较。问题是，日志文件因为新记录的添加而不断增长，从而导致完整性检查的削弱。为了解决这一矛盾，应该注意完整性监控只能保证没有被日志记录组件不断写入的日志的完整性。

10.6

下一个需求最为重要，也是最容易被忽视的。许多PCI实施者忘记了，PCI需求10不仅要求"有日志"，还要求"有日志，并且查看它们。"10.6小节明确指出PCI组织应该按照PCI DSS，"至少每天一次审核所有系统组件的日志。日志审核必须至少包含执行IDS和AAA服务器（例如RADIUS）等安全功能的服务器"（《Requirements and Security Assessment Procedures》，2010）。本章余下的部分介绍详细的日志审核规程和常规做法。

因此，PCI DSS需求还覆盖了需要"每天审核"的日志源范围，而不仅是需要配置日志、保留或者集中化日志。考虑到大型IT环境每天可能生成以GB计算的日志，人工阅读所有日志是不可能的。因此，PCI DSS在这个需求上有个注释："可以使用日志收集、解析和警报工

具实现需求 10.6 的依从性”(《Requirements and Security Assessment Procedures》，2010)。

10.7

最后一个需求(10.7)处理另一个非常重要的日志记录问题——日志留存，它规定：“留存审计跟踪历史至少一年，保持在线可用性至少三个月”(《Requirements and Security Assessment Procedures》，2010)。和其他无数需求不同，这一需求直接处理复杂的日志留存问题。因此，如果你无法查看 1 年前的日志，就违反了规定。而且，PCI DSS 在更新后的版本 1.1 中还明确地增加了一年的需求规定。

下面，我们来总结一下目前为止学习的 PCI 日志记录需求：

- PCI 需求 10 规定以预先定义的细节水平记录来自范围内所有系统的特定事件。
- PCI 要求日志中所有操作与实际用户挂钩。
- 范围内的系统上的时钟和时间应该同步。
- 应该保护所有收集日志的 CIA(机密性、完整性、可用性)。
- 日志应该定期审核；特定的日志应该至少一天审核一次。
- 范围内的所有日志应该留存至少一年。

现在，我们做好准备，将要更加深入地研究日志和监控，这些日志和监控不仅在需求 10 中规定，还出现在所有其他 PCI 需求中。许多人可能认为 PCI 中的日志都在需求 10 中说明，但实际情况更为复杂：日志出现和隐藏在所有其他的小节中。

16.4.2 与日志记录相关的其他需求

几乎所有满足 PCI DSS 需求的规定(如数据加密或者防病毒更新)，都可能使用日志文件来证明。

例如，需求 1“安装和维护防火墙配置，以保护持卡人数据”(《Requirements and Security Assessment Procedures》，2010)中提到，组织必须有“批准和测试所有外部网络连接以及防火墙配置变更的正式过程”(《Requirements and Security Assessment Procedures》，2010)。但是，在建立这种过程之后，人们必须验证防火墙配置变更确实得到授权，并且按照书面的变更管理规程进行。这时日志记录变得极其方便，因为它说明了实际发生的情况，而不是“应该发生”的情况。

需求 1.3 包含防火墙配置的指导原则，有关于入站和出站连接性的具体陈述。必须使用防火墙日志来验证这些配置，仅仅审核配置还不够，因为只有日志才能说明“实际发生了什么”而不只是“如何配置”。

类似地，需求 2 讨论密码管理“最佳实践”和一般的安全加固，例如不运行没有必要的服务。日志可以说明何时启动了之前禁用的服务，这可能是不知情的系统管理员或者攻击者的操作造成的。

需求 3 更进一步，处理数据加密，这和日志记录有着直接而明确的联系。例如，3.6 小

节（后文将以简略的形式展示）隐含了一个要求：拥有验证这些活动是否真正进行的日志。确切地说，大部分加密系统记录密钥生成、分发和废止的日志，这些日志对于满足上述需求是至关重要的。

需求 4 处理的也是加密，由于同样的原因，和日志记录也有隐含的联系。

需求 5 针对病毒防御。当然，为了满足 5.2 小节的需求——“确保所有防病毒机制最新、活跃地运行，并且能够生成审计日志”（《 Requirements and Security Assessment Procedures 》，2010），必须查看所提到的这些日志。

所以，即使是“使用和定期更新防病毒软件”也可能在评估期间需要请求日志数据，因为这些信息由防病毒评估日志提供。著名的“防病毒更新失败”问题也在日志中反映，它说明公司正处于恶意软件的威胁之下，因为没有更新最新特征码的防病毒软件只能造成虚假的安全感，损害依从性工作。

需求 6 也处于同一范畴：它要求组织“开发和维护安全的系统和应用程序”（《 Requirements and Security Assessment Procedures 》，2010），没有强大的评估日志记录功能和应用程序安全监控，这是无法想象的。

需求 7 规定“根据业务按需了解的原则限制对持卡人数据的访问”（《 Requirements and Security Assessment Procedures 》，2010），需要日志来验证谁能够访问文中所指的数据。如果不应该看到数据的用户出现在访问数据的日志文件中，就需要采取纠正措施。

为每个访问系统的用户分配唯一的 ID 符合其他安全“最佳实践”。在 PCI 中，这不仅是一个“最佳实践”，还是一个需求（需求 8“为每个具有计算机访问权的个人分配一个唯一的 ID”，《 Requirements and Security Assessment Procedures 》，2010）。显然，人们必须“控制用户 ID、证书和其他标识符对象的添加、删除和修改”（PCI DSS 8.5.1 小节）。大部分系统记录这些活动的日志。

除此之外，8.5.9 小节“每 90 天至少更改用户密码一次”（《 Requirements and Security Assessment Procedures 》，2010），也可以通过审核服务器日志来验证，以便确保所有账户在 90 天内至少修改一次密码。

需求 9 介绍了安全的一个新领域——物理访问控制。甚至连包含来访者日志维护（可能以物理日志簿的形式）的 9.4 小节也和日志管理有关（如果这些访问者日志是电子形式的）。对于这类日志有单独的数据留存需求：“使用访问者日志维护访问者活动的物理评估跟踪。留存这些日志至少 3 个月，除非法律有其他限制”（《 Requirements and Security Assessment Procedures 》，2010）。

需求 11 说明扫描（或者“测试”）范围内系统漏洞的必要性。但是，在 11.4 小节中要求使用 IDS 或者 IPS：“使用网络入侵检测系统、基于主机的入侵检测系统和入侵预防系统，监控所有网络流量，并向有关人员发出可疑入侵的警报。保持所有入侵检测和预防引擎最新”（《 Requirements and Security Assessment Procedures 》，2010）。入侵检测只在审核日志和警报时才有用。

需求 12 涵盖了更高级的问题——安全策略、安全标准和日常运营规程（例如，需求 10

中强制的每日日志审核规程应该在这里反映）。但是，它也和日志记录相关，因为评估日志记录应该是每个安全策略的一部分。除此之外，事故响应需求也和日志记录相关："建立、记录和发布安全事故响应和升级规程，确保及时和有效地处理所有情况"（《Requirements and Security Assessment Procedures》，2010），如果没有高效的日志数据收集和及时审核，上述要求难以想象。

因此，PCI DSS 中的事件日志记录和安全监控远不止于需求 10。只有通过精心的数据收集和分析，才能满足 PCI 的广泛需求。

下一小节详细讨论日志记录需求，重点放在需求 10 和 12。

16.5　日志记录策略

根据 PCI DSS 需求，源于 PCI 的日志记录策略至少必须包含如下内容：

- 足够的日志记录，包括日志事件类型（登录 / 注销、资源访问、防火墙接受 / 拒绝、IPS/IDS 警报等）和细节。
- 日志聚合和留存（1 年）。
- 日志保护（确保日志不被篡改）。
- 日志审核

日志记录策略定义应该捕捉日志数据的哪些属性供以后审核、升级和响应。同样，本章中使用 PCI DSS 的概念，但是这个策略是通用的，可以应用于非 PCI 活动、应用程序等场合。

现在，我们将重点放在日志审核上。PCI DSS 指出："至少每天一次审核所有系统组件的日志。日志审核必须包含执行入侵检测系统（IDS）和身份认证、授权等安全功能的服务器，以及计费协议服务器（例如 RADIUS）"（《Requirements and Security Assessment Procedures》，2010）。然后，它又增加了这一条："可以使用日志收集、解析和警报工具满足需求 10.6 的依从性"（《Requirements and Security Assessment Procedures》，2010）。PCI 日志审核测试和验证规程强制，合格安全评估员（QSA）应该"获得和检查安全策略及规程，验证它们包含每天至少一次审核安全日志的规程，并且要求对异常情况进行后续调查"（《Requirements and Security Assessment Procedures》，2010）。QSA 还必须确定"通过观察和面谈，验证所有系统组件都进行了定期日志审核"（《Requirements and Security Assessment Procedures》，2010）。

注释　QSA

合格安全评估员（QSA）是对处理信用卡数据的企业进行 PCI 依从性评估的合格（通过培训、认证等）人员。

最后，在下一个小节，我们讨论日志审核规程的应用和工作流程，包括：

1）日志审核方法、模式和任务。

2）异常调查和分析。

3）规程和管理报告的验证。

这些规程可以使用自动化的日志管理工具，当这些工具不可用或者不兼容应用程序生成的日志格式时，也可以人工进行。

16.6 审核、响应、升级规程

三种 PCI 强制规程之间的联系如图 16.1 所示。

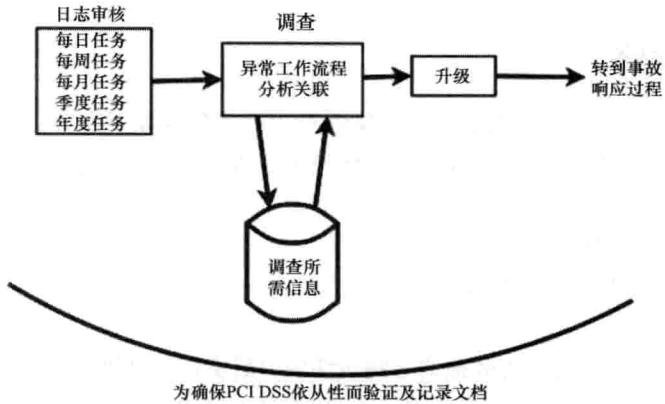


图 16.1 确保 PCI DSS 依从性

换句话说，每天进行“定期日志审核工作”（如果每天审核不可能，可以选择较低的频率），发现的任何异常情况被升级到“异常调查和分析”。这两者都按照本章的“日志审核验证”小节中的规定记录文档，建立依从性证据。我们现在将提供这三种任务的细节。

16.6.1 定期日志审核方法和模式

本小节介绍日志管理的定期日志审核规程。这种审核由应用程序管理员或者安全管理员（见 16.3 小节）进行，可以使用自动化工具（在 PCI DSS 中明确允许），如果这些自动化工具不可用或者不支持 PCI 应用程序中的日志类型，则人工进行。

我们将为两种情况建立完整的端到端规程，然后用例子加以说明。

PCI DSS 定期日志审核（尽管它可能没有每天对所有应用程序进行，但是仍被称为“每天日志审核”）的基本原则是实现如下目标：

- 确定持卡人数据没有遭到攻击者的入侵。
- 尽早检测出持卡人数据的可能风险。
- 满足 PCI DSS 日志审核的明确需求。

PCI DSS 是每天日志审核的动机，但是它还能实现其他的目标：

- 确保处理持卡人数据的系统安全有效地运行。
- 协调日志中观察到的与其他系统和活动之间所有可能的异常情况（例如应用程序代码变化或者补丁部署）。

根据上述目标，每天日志审核围绕“基线”或者常规日志消息集的学习和文档记录构建。基线的讨论参见第 10 章。

然后，按照基线，从常规例程中寻找“异常”，并调查它们，以确保没有发生（或者即将发生）持卡人数据的泄露。

图 16.2 说明了这一过程。

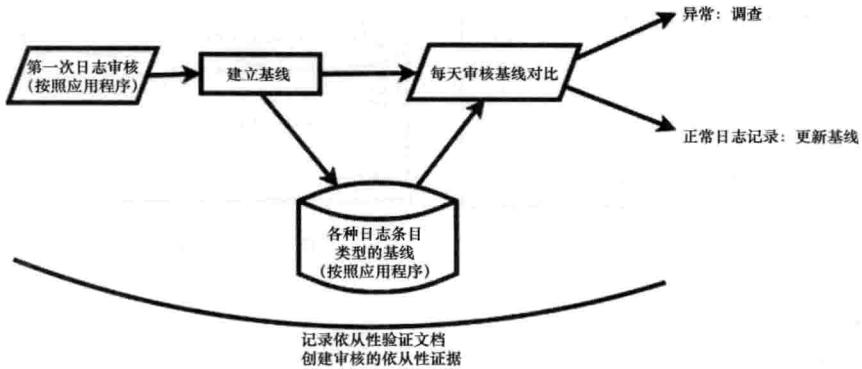


图 16.2　基线过程

在进行 PCI 每天日志审核之前，熟悉每个应用程序在日志中记录的正常活动是关键（Anton Chuvakin 博士的博客，2011）。

构建的主要基线包括日志消息类型，如图 16.3 所示。

图 16.3 是第一次出现，我们确定该消息不表示持卡人数据安全的关键故障，我们可以将其添加到预期的基线。图 16.4 展示了一个 Windows 7 事件 ID 示例。

需要注意的是，有些日志类型中可能没有明确的事件类型。例如，有些 Java 应用程序日志和 Unix 日志中没有记录明确的日志或者事件类型。所需要的是创建一个隐含的事件类型（《Requirements and Security Assessment Procedures》，2010）。

这种情况的规程如下：

1）审核日志消息。

2）识别日志消息的哪一部分说明了相关的情况。

3）确定消息的这一部分是不是唯一的。

4）从消息的这一部分创建一个事件 ID。

尽管日志管理工具自动进行这一过程，但是在采用人工日志审核规程时，研究一个人工的例子是很有意义的。例如（Anton Chuvakin 博士的博客，2011）：

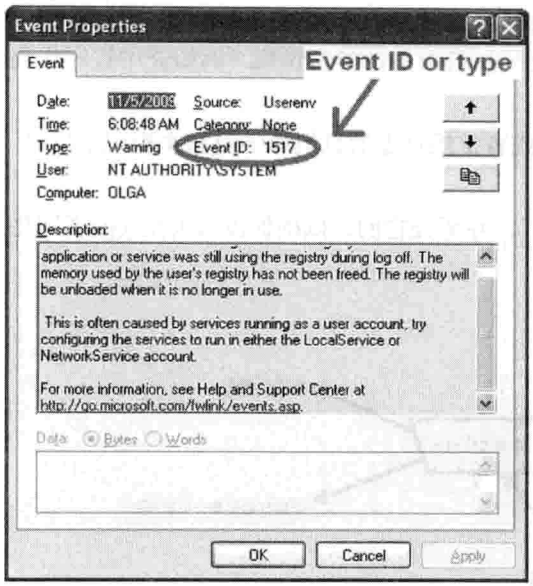


图 16.3　Windows XP 事件 ID 示例

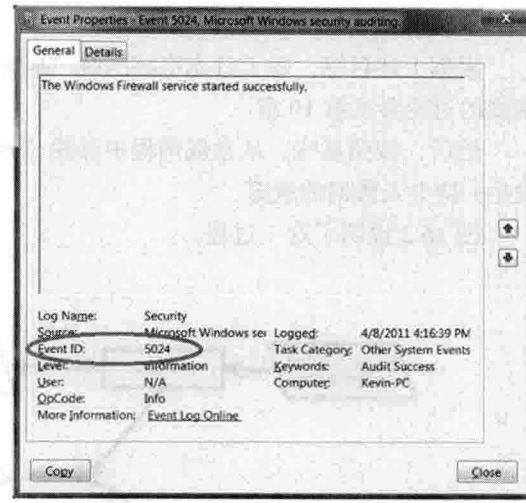


图 16.4　Windows 7 事件 ID 示例

示例 1

1）审核日志消息

日志消息如下：

```
[Mon Jan 26 22:55:37 2004] [notice] Digest: generating secret for
   digest authentication.
```

2）识别消息的哪一部分说明了相关的情况

消息的关键部分很可能是“generating secret for digest authentication”(生成摘要身份认证的秘密”，甚至是“generating secret”。

3）确定消息的这一部分是不是唯一的

对日志中其他消息的审核说明，没有其他消息包含相同的短句，因此这一短句可以用于分类这一条消息。

4）从消息的这个部分创建事件 ID

我们可以创建一个事件 ID 或者消息类型“generating_secret”。现在我们可以用观测到的这类消息更新基线。

下面我们介绍另一个 Java 应用程序日志的示例。

示例 2

1）审核日志消息

日志消息如下：

```
>>>>17 Jul 2008
```

```
13:00:57|AFWVGBYMBHWMV|ERROR|MerchantPreferredQuoteStrategy| Error
getting merchant availability, will fall back to XFS
com.factory.service.travel.flying.exception.MerchantSupplierException
   at com.
factory.service.travel.flying.PropertyAvailServiceImpl.
execute(MachineryAvailServiceImpl.java:407)
   at
<truncated>
```

2）确定消息的哪一部分说明了相关的情况

消息的关键部分很可能是“Error getting merchant availability”或者“Error getting availability”

3）确定消息的这一部分是不是唯一的

对日志中其他消息的审核发现，没有其他消息包含短句“Error getting merchant availability”，因此可以使用它分类这条消息。但是，其他消息包含短句“Error getting object availability”，因此不建议使用“Error getting merchant availability”作为消息类型（Anton Chuvakin 博士的博客，2011）。

4）从消息的这个部分创建事件 ID

我们可以创建一个消息 ID 或者消息类型“Error_getting_merchant_availability”。现在我们可以用观测到的这类消息更新基线。

可以使用如下过程快速构建初始基线，包括两种情况：使用自动化的日志管理工具和不使用自动化工具。

除了这个“事件类型”之外，对前一天（过去 24 个小时）重叠的日志条目容量进行快速评估也很有意义。日志容量的显著不同也应该用下面定义的规程进行调查。特别是，日志记录丢失（常常从日志条目容量的明显减小中发现）必须加以调查，并升级为安全事故（Anton Chuvakin 博士的博客，2011）。

16.6.2 用日志管理工具构建一个初始基线

执行如下步骤，用日志管理工具构建一个基线：

1）确保日志管理工具收集 PCI 应用程序中的相关日志（Chuvakin，日期不详）。

2）确定工具能够“理解”（解析、标记等）消息，并标识每个日志的“事件 ID”或者消息类型（Chuvakin，日期不详）。

3）为初始基线选择一个时间周期：“90 天”或者在日志收集不到 90 天时“全时”（Complete PCI DSS Log Review Procedures Part 8，2011）。

4）运行一个报告，显示每个消息类型的计数。这个报告指出系统运营 90 天内遇到的所有日志类型（Chuvakin，日期不详）。

5）如果没有发现任何卡数据泄露，我们可以接受如上报告，作为“例行运营”基线（Anton Chuvakin，2011）。

6）创建基线的同时应该执行一个附加步骤：即使我们断定卡数据没有遭到入侵，在 90 天内记录的某些日志消息也可能触发某种行动或者补救措施。这样的消息称为“已知错误”，应该标记出来（Complete PCI DSS Log Review Procedures Part 8，2011）。

让我们来看一个上述策略的完整示例：

1）确保日志管理工具收集 PCI 应用程序中的相关日志。

在这一步中，我们观察日志管理工具，验证来自 PCI 应用程序的日志。可以通过查看所有日志记录设备的报告来完成这一步。

表 16.2 表明必要的收集已执行。

表 16.2　所有日志记录设备的报告

设备类型	设备名称	日志消息数
Windows 2003	Winserver1	215 762
Windows 2003	Winserver2	215 756
CEMS	CEMS1	53 445
TOPPS	Topps1server	566
TABS	Tabs	3 334 444

注：时限为 2009 年 1 月 1 日 ~ 3 月 31 日（90 天）

2）确定工具能够“理解”（解析、标记等）消息，并标识每个日志的“事件 ID”或者消息类型（Chuvakin，日期不详）。

这一步通过比较工具里的消息计数（例如上面显示日志消息计数的报告）和原始日志中的消息计数来完成。

3）为初始基线选择一个时间周期：“90 天”或者在日志收集不到 90 天时“全时”。

在这个例子中，因为有足够的可用日志，我们选择 90 天。

4）运行一个报告，显示每个消息类型的计数。例如，如表 16.3 所示的报告。

表 16.3 中的报告指出系统运营 90 天内遇到的所有日志类型。

表 16.3　显示消息类型计数的报告

事件 ID	事件描述	计　数	平均计数 / 天
1517	注册表故障	212	2.3
562	登录失败	200	2.2
563	登录成功	24	0.3
550	用户证书 218 更新	12	0.1

注：时限为 2009 年 1 月 1 日 ~ 3 月 31 日（90 天）

5）如果没有发现任何卡数据泄露，我们可以接受如上报告，作为“例行运营”基线。

在第一次日志审核期间，接受其作为正常基线之前可能有必要调查一些日志事件。下一步介绍如何完成这种调查。

6）创建基线的同时应该执行一个附加步骤：即使我们断定卡数据没有遭到入侵，在 90

天内记录的某些日志消息也可能触发某种行动或者补救措施。这样的消息被称作“已知错误”，应该标记出来（Complete PCI DSS Log Review Procedures Part 8，2011)。

90 天摘要中的某些日志实际上表示问题发生，需要调查（见表 16.4）。

表 16.4　90 天摘要报告

事件 ID	事件描述	计　数	平均计数 / 天	正常或者“错误”
1517	注册表故障	212	2.3	
562	登录失败	200	2.2	
563	登录成功	24	0.3	
550	用户证书更新	12	0.1	
666	内存耗尽	1	N/A	行动：重启系统

在这个报告中，我们注意到最后一行事件 ID=666 的日志记录，事件名称为“内存耗尽”的日志记录在 90 天中只发生了一次。这种罕见的事件至少应该引起人们的兴趣；消息的描述（内存耗尽）可能代表着一个潜在的严重问题，因此需要按照下面介绍的调查规程进行调查。

人工创建一个基线是可能的，但是较为复杂。我们将在下一小节探索这一任务。

16.6.3　人工构建初始基线

在日志不兼容可用工具或者可用工具不能很好地理解日志数据时，必须在不使用日志管理工具的情况下人工构建基线。为此，执行如下步骤（Complete PCI DSS Log Review Procedures Part 8，2011）：

1）确保来自 PCI 应用程序的相关日志保存在某个位置。

2）为初始基线选择一个时间周期：“90 天”或者在日志收集不到 90 天时“全时”；检查最早的日志时间戳确定这一点（Complete PCI DSS Log Review Procedures Part 8，2011）。

3）从最旧的日志条目开始审核，直到最新的条目，尝试识别它们的类型 (Complete PCI DSS Log Review Procedures Part 8，2011)。

4）人工创建所有观测类型的摘要；如果现实的话，收集每条消息出现的次数（在日志数据量很大的情况下不可能；Complete PCI DSS Log Review Procedures Part 8，2011）。

5）如果没有发现任何卡数据泄露，我们可以接受如上报告，作为“例行运营”基线 Complete PCI DSS Log Review Procedures Part 8，2011)。

6）创建基线的同时应该执行一个附加步骤：即使我们断定卡数据没有遭到入侵，在 90 天内记录的某些日志消息也可能触发某种行动或者补救措施。这样的消息被称作“已知错误”，应该标记出来（Complete PCI DSS Log Review Procedures Part 8，2011)。

确定“已知错误”消息的原则

下面是创建基线的过程中将某些消息标记为“已知错误”的粗略原则。如果生成了这些消息，在每日审核过程中应该首先观察：

1）在不寻常的时间发生的登录和其他“授权访问”日志消息。(从技术上说，为了更高的准确性，这也需要创建一个基线。但是，发生在业务时间（在正确的时区中）之外的登录通常至少是令人感兴趣的。)

2）发生在更改窗口之外的证书和权限修改日志消息。

3）到期的用户账户生成的任何日志消息。

4）维护窗口（如果作了定义）之外的重启消息。

5）在备份窗口（如果作了定义）之外进行的数据备份 / 导出。

6）日志数据删除。

7）系统或者应用程序日志记录终止。

8）系统或者应用程序日志记录配置的任何变更。

9）在过去触发了任何行动（系统配置、调查等等）的日志消息。

10）与安全策略违规明显相关的其他日志。

正如我们所看到的，这个列表对于创建“以近乎实时的方式监控什么？”的策略非常有用，而不仅仅适用于日志记录。随着时间的推移，该列表应该根据本地应用程序日志的知识和过去调查中了解到的情况进行扩展。

构建初始基线之后，我们就可以开始每天日志审核。

16.6.4 主要工作流程：每天日志审核

这是日志审核中非常核心的部分——将最后一天产生的日志（每日审核的情况下）与积累的基线相比较。

遵循这个模型的每日工作流程如图 16.5 所示。

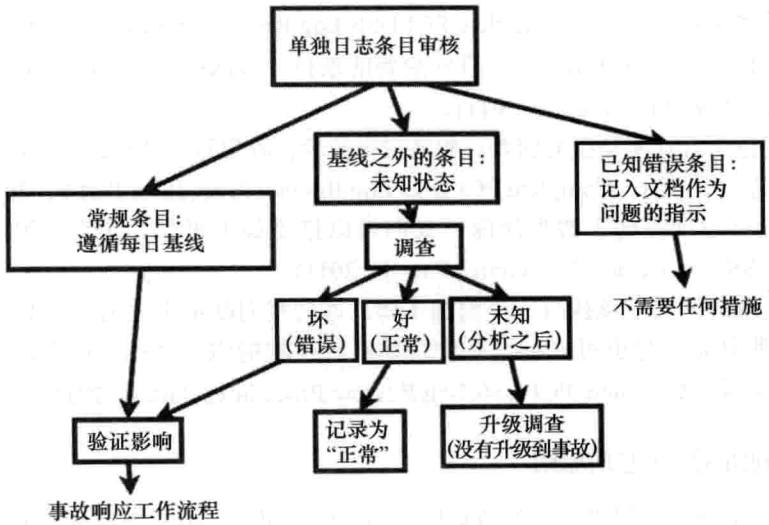


图 16.5　日常工作流程

该图总结了执行每天日志审核的日志分析人员的行为。在继续之前，必须处理日志审核频率的问题。

定期日志审核的频率

PCI DSS 需求 10.6 明确规定，“每天至少一次审核所有系统组件的日志”，假定日志审核规程每天进行（Anton Chuvakin 博士的博客，2011）。只有 QSA 可以根据用于补偿性控制的相同原则，批准你采用较低的日志审核频率。批准较低的日志审核频率需要什么理由？下面的列表包含了日志审核执行频率低于每天一次的某些理由：

- 应用程序或者系统没有每天生成日志。如果日志记录没有每天添加，就没有必要每天进行日志审核。
- 日志审核使用以批模式收集日志的日志管理系统，日志批次到达频率低于每天一次。
- 应用程序没有处理或者存储信用卡数据；只因为直接连接而被列入范围（Anton Chuvakin，2011）。

记住，只有你的 QSA 的意见有约束力，其他人都没有！

如何真正地将今天的日志批次与基线比较？可以采用两种方法，它们都广泛地用于日志审核——根据可用资源和使用的工具作出选择，如图 16.6 所示。

在这两种方法中，第一种方法只考虑之前没有观测到的日志类型，可以人工进行，也可以使用工具进行。尽管它很简单，但是对于许多日志类型却极其有效：只注意生成的新日志消息类型，通常非常有助于深入了解安全、依从性和运营（Anton Chuvakin 博士的博客，2011）。

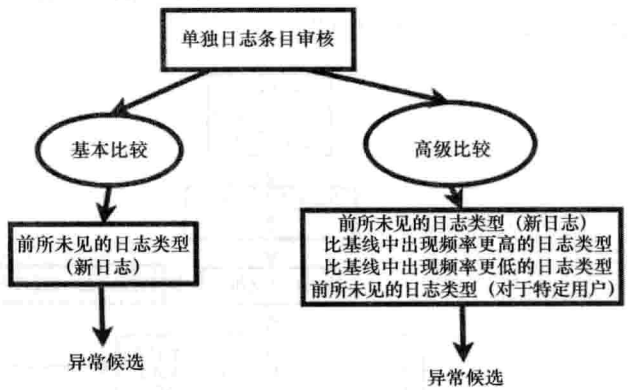


图 16.6　选择异常候选者

例如，如果 ID 为 1-7 的日志消息每天都大量产生，但是 ID 为 8 的日志却从未出现，这种日志消息的每次出现就成为调查的理由。如果确定该消息是正常的，不会触发任何行动，以后可以将其添加到基线中。

每天日志审核的比较方法总结如下：

- 基本方法：
  - 前所未见的日志类型（新日志消息类型）。
- 高级方法：
  - 前所未见的日志类型（新日志消息类型）。
  - 比基线中出现频率更高的日志类型

- 比基线中出现频率更低的日志类型
- 前所未见的日志类型（对于特定用户）。
- 前所未见的日志类型（对于特定应用程序模块）。
- 前所未见的日志类型（在周末）。
- 前所未见的日志类型（在工作日内）。
- 记录的新用户活动（系统中前所未见的用户的任何日志）。

遵循高级方法的同时，日志管理工具也可以使用其他比较算法。

在消息被标记为异常之后，我们转入日常工作流的不同阶段——从每天审核转向调查和分析。

16.6.5 异常调查与分析

不匹配正常配置的消息被标记为“异常”。需要注意的是，异常不等同于安全事故，但可能是安全事故发生的早期指示。

在这一阶段，我们有了一条落在例行 / 正常运营之外的日志消息。我们该如何确定它是否严重，确定其对安全和 PCI 依从性状态的影响呢？（Anton Chuvakin 博士的博客，2011）。

1. 初始调查

图 16.7 展示了对每个“异常”条目使用的调查过程（“初始调查”）的概要情况（本章还将进一步地介绍更多的细节）。

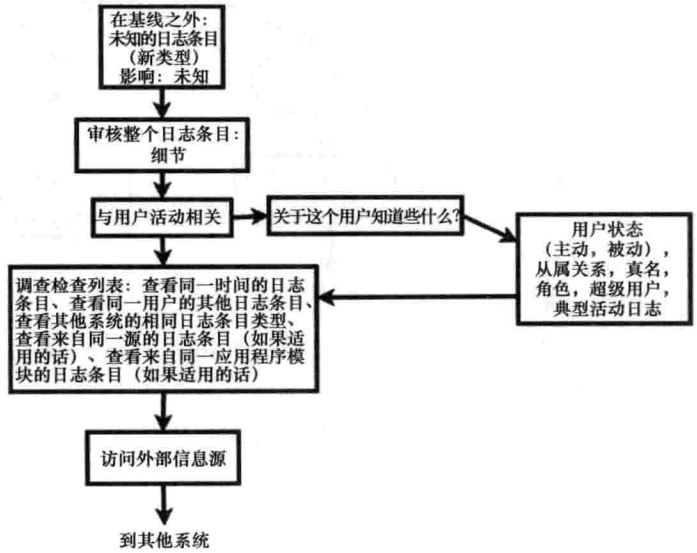


图 16.7　每个异常情况的调查

确切地说，这展示了一个使用日志调查检查列表的过程，下面将详细地介绍这个检查列表：

1）查看同一时间的日志条目：这一技术涉及增加围绕所调查日志消息的时间范围。大

部分日志管理产品能够审核或者搜索特定时间范围内的所有日志。例如：

a）首先，查看“可疑日志”前后 1 分钟触发的其他日志。

b）其次，查看“可疑日志”前后 10 分钟触发的其他日志。

c）最后，查看“可疑日志”前后 1 小时触发的其他日志。

2）查看来自同一用户的其他条目：这一技术包括搜索同一用户的活动生成的其他日志条目。常常发生这种情况，用户活动的特定日志事件只能在同一用户其他活动的上下文中解读。大部分日志管理产品允许你“下钻”或者搜索特定时间范围内的特定用户。

3）查看其他系统上的同类条目：这种方法包括搜索其他同类日志消息，但是在不同的系统上进行，以确定其影响。了解其他系统上何时产生相同的消息，可以得到理解日志消息影响的线索。

4）查看来自同一源的条目（如果适用的话）：这种方法涉及审核来自网络相关源地址的其他日志消息。

5）查看来自同一应用程序模块的条目（如果适用的话）：这种方法涉及审核来自同一个应用程序模块或者组件的所有其他日志消息。同一时间范围的其他消息（见上面的第一条）可能很重要，但是审核来自相同组件的最近日志一般有助于揭示真实的情况。

在某种情况下，上述检查列表可能没有作用。也就是说，异常日志条目对安全和 PCI 依从性的影响仍然未知。在这种情况下，我们必须从其他系统获得信息，例如文件完整性监控（FIM）、补丁管理（PM）、变更管理（CM）和其他系统（Chuvakin，2011）。

2. 外部信息源调查

调查规程可以扩展，覆盖组织的其他可用信息源。图 16.8 展示了在调查中使用外部信息

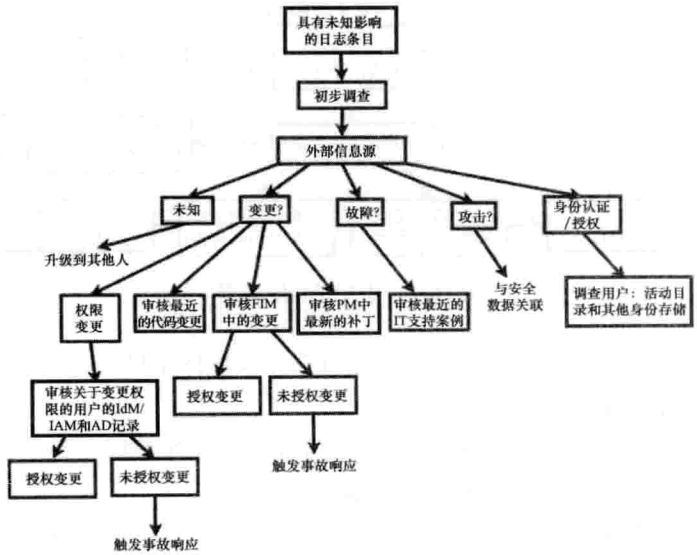


图 16.8　在调查中使用外部信息

源所遵循的规程。

这一规程的主要思路是根据异常日志条目的类型来识别并查询信息源，然后识别其影响和必要的行动（如果有的话）。

该规程粗略地确定日志条目的类型，然后查询相关的信息源。在某些情况下，这种日志条目确实是严重问题的征兆，会触发事故响应过程。

但是，有时候初步分析和外部系统查询都不能得到结果，这样的“异常”日志很罕见。在这种情况下，触发协作工作流（Chuvakin 博客，2011）。

3. 升级到其他规程

本规程可以利用组织中其他人的知识，这些人可能了解“异常”情况。调查和升级过程如图 16.9 所示。

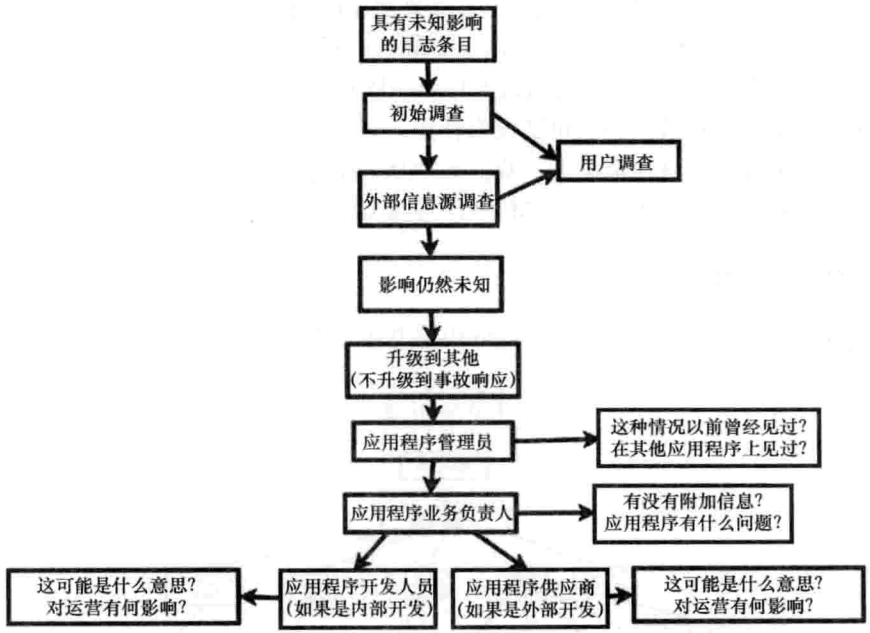


图 16.9　升级到其他规程

这一规程的主要思路是找出可能了解应用程序中发生的事件的合适人员，并与之面谈，确定事件的影响和必要的行动（如果有的话）。

最后的资源是咨询应用程序供应商，这种信息请求通常很费时，甚至很昂贵（取决于支持合同），所以应该审慎使用。（Chuvakin，日期不详）

在进入下一部分之前，我们需要讨论事故响应和升级，下一小节提供了这些主题的更多细节。

16.6.6　事故响应和升级

迄今为止介绍的许多工作流依赖事故响应（IR）和升级。但是，什么是事故响应和升级呢？本节提供 IR 和升级如何融入本章介绍的整体过程的必要理解。你将能很好地理解这一点。

我们首先讨论升级。在日志管理和 IT 的一般概念中，升级规程用于确定何时应该升级所发生的情况。

图 16.10 展示了 SANS 定义的 IR 过程。

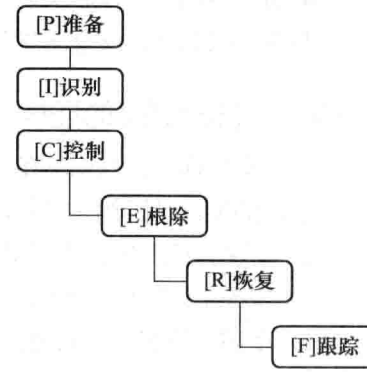


图 16.10　SANS IR 过程

我们来讨论这个过程的每一步。

准备阶段包括事故之前必须完成的任务：从组建团队、训练人员、收集和构建工具，到部署附加监控和创建过程及事故规程。这也是你开发升级规程 / 策略的时机。这一策略是关于合适的个人、团体之间沟通的。（Anton Chuvakin，2011）。

识别阶段从事故的征兆被发现开始。这不仅和 IDS 警报有关（实际上这种情况也不太多），还与可能揭示事故出现的各种信号的分析有关。例如，持续的 CPU 利用、系统重启和维护窗口之外的配置更改等，都是说明发生了某些值得进一步注意的事件的信号。

控制阶段对于在文档中记录发生的情况、隔离受影响系统（甚至区域）以及进行著名的“拔线”操作（也就是断开或者关闭系统）来说很重要。在这一阶段，我们还试图审核其他系统可能受到的影响和损害，评估破坏范围。

根除阶段中，我们评估可用备份，准备恢复或者重建系统，为回到正常状态做准备。这一阶段还要规划网络的更改。

恢复阶段是一切回到正常的阶段。在此阶段要起用额外的日志记录和监控。

在跟踪阶段，你讨论和记录得到的教训，向管理层报告事故，并规划未来事故处理中的改进。这一阶段也常称为后续阶段。

16.7　日志审核的验证

日志审核最后的关键部分是确保有足够的证据，证明这一过程被真正严格地实施。好消息是，可以使用日志记录和日志审核过程管理报告中的相同数据。下面我们来确定应该制作哪些文档，作为日志审核的证据。

首先，常见的错误概念是用实际的日志来提供这种证明。这并不正确：“有日志”和“审核了日志”是完全不同的概念，有时候，需要对安全和依从性计划进行多年的成熟化，才能分辨这两者。

只需要记住，在 PCI DSS 依从性验证中，我们需要证明多个主要的部分。下面是需要组合的所有依从性证据的总列表。和其他小节不同，在此我们将包含日志记录的证据，而不只是日志审核的证据，因为后者依赖于前者：

- 日志记录的存在和充足性。
- 日志审核过程的存在及其实施。
- 异常处理过程及其实施。

现在，我们可以围绕这些领域组织证据，然后构建过程来收集这些证据。

16.7.1 日志记录的证据

第一类证据是日志记录存在和充足性的证据。这一部分是三者中最容易证明的。

下面的项目可以作为日志记录的证据：

1）有文档证明的日志记录策略，包括记录的事件和每个事件记录的细节。

2）实施上述策略的系统 / 应用程序配置文件。

3）上述应用程序生成的，符合策略的日志。

16.7.2 日志审核的证据

第二类：日志审核过程及其实施的证据。这一部分比前一部分更难证明。

下面是可以作为日志审核证据的项目：

1）有文档证明的日志记录策略，涵盖日志审核。

2）有文档证明的操作规程，详细说明日志审核的步骤。

3）相关人员执行的日志审核任务记录（有些日志管理产品创建一个审核报告和事件的审计记录；这种审计跟踪应该涵盖，下面会介绍人工审核的情况），这个项目可以视为“日志审核日志”。

4）调查的异常情况记录（下一小节将作介绍）也间接地证明日志审核的进行。

16.7.3 异常处理的证据

第三类：异常处理过程及其实施的证据。这一部分是目前为止最难以证明的。

下面的项目可以作为日志异常处理的证据：

1）有文档证明的日志记录策略，涵盖异常及其处理。

2）有文档证明的操作规程，详细说明日志审核中找到的异常的调查步骤。

3）所有异常调查及所采取的行动的日志（“日志簿”）（Complete PCI DSS Log Review Procedures Part 8，2011）。

上述的证据应该能为组织认真遵循 PCI DSS 原则提供充足的证据。我们将重点放在这一证据的产生上，表 16.5 对此作了详细的介绍。

表 16.5　依从性证据

PCI 依从性日志记录子域	依从性证据	如何获取证据
日志记录存在和充足的证据	有文档证明的日志记录策略	如果不存在策略，则创建
日志记录存在和充足的证据	系统 / 应用程序配置文件	部署之后，保存配置文件作为主拷贝
日志记录存在和充足的证据	上述应用程序生成的日志	收集日志样本，保存为依从性的证据
日志审核的证据	有文档证明的日志记录策略	如果不存在策略，则创建
日志审核的证据	有文档证明的操作规程	根据本章的工作流和规程创建
日志审核的证据	执行的日志审核任务记录	使用工具或者创建一个“日志簿”(格式见下文)
日志审核的证据	所调查的异常情况的记录	创建一个调查“日志簿”
异常处理的证据	有文档证明的日志记录策略	如果不存在策略，则创建
异常处理的证据	有文档证明的操作规程	根据本章的工作流和规程创建
日志审核的证据	所调查的所有异常情况的记录	创建一个调查 " 日志簿“或者知识库

上述列表中关键的项目是“日志簿”，它用于记录异常的跟踪和调查，从而创建了 PCI DSS 需求依从性的有力证据。日志簿甚至可以成长为更加高级的形式——包含过去所有异常分析案例的调查“知识库”。

16.8　日志簿——异常调查的证据

如何创建一个日志簿？日志簿用于记录与日常审核期间标记的异常的分析和调查相关的所有情况。虽然事故处理过程（如 SANS 事故响应工作流）中使用相同的日志簿方法，但在这里我们将其当作依从性的证据。

日志簿应该记录所有涉及的系统，所有面谈过的人，采取的所有行动及其理由，得到的结果，使用的工具和命令（及其结果）等。

下一小节详细介绍日志簿条目的内容。

16.8.1　日志簿推荐格式

日志簿条目应该包含如下内容：

1）日志簿条目开始的日期 / 时间 / 时区。

2）启动日志簿条目的人的姓名和角色。

3）启动该条目的理由：记录异常（从日志聚合工具或者原始日志文件中复制而来）、确保复制整个日志，尤其是时间戳（可能不同于记录时间）和来源系统（什么 / 何时 / 何地，等等）。

4）详细说明日志为什么不正常，为什么进行这项分析。

5）产生异常日志记录或者与日志异常相关的系统的相关情况（Complete PCI DSS Log Review Procedures Part 8，2011）：

a）主机名

b）操作系统

c）应用程序名

d）IP 地址

e）位置

f）所有权（如果知道的话）

g）系统关键性（如果已经定义且适用）

h）在补丁管理、变更管理、FIM 等过程中。

6）产生该日志的活动的用户相关信息（如果适用）。

7）遵循的调查规程、使用的工具、屏幕截图等。

8）采取的调查行动。

9）日志分析过程中接触的人。

10）在分析过程中确定的影响。

11）行动和缓解措施的建议（如果必要；Anton Chuvakin 博士的博客，2011）。

16.8.2 日志簿条目示例

下面是遵循上述模式的一个例子，使用了图 16.11 中的日志条目：

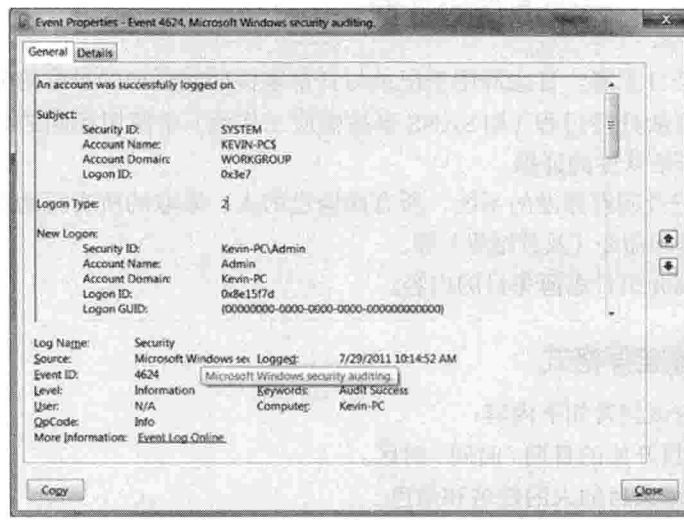


图 16.11　日志簿示例使用的 Windos 7 事件

1）日志条目启动的日期 / 时间 / 时区：2011 年 6 月 30 日，下午 4 点 15 分，美国东部时间。

2）启动日志簿条目的人的姓名和角色：Kevin Schmidt。

3）启动该条目的理由：记录异常（从日志聚合工具或者原始日志文件中复制而来）、确保复制整个日志，尤其是时间戳（可能不同于记录时间）和来源系统（什么 / 何时 / 何地，等等）。

日志时间 / 日期：2011 年 7 月 29 日，上午 10 点 14 分 52 秒，美国东部时间。

系统：Kevin-PC.example.com。

4）详细说明日志为什么不正常，为什么进行这项分析。

来自这个源（Microsoft Windows 安全审计）的事件 ID（Windows 事件 ID 4624，成功的账户登录）以前从未在审核日志所在的任何系统上发现过。

5）产生异常日志记录或者与日志异常相关的系统的相关情况：

a）主机名：Kevin-PC.example.com

b）操作系统：Windows 7

c）应用程序名：无

d）IP 地址：10.1.1.1

e）位置：总公司

f）所有权（如果已知）：Kevin Schmidt

g）系统关键性（如果定义并适用）：关键，负责人的便携电脑。

h）在补丁管理、变更管理、FIM 等过程中：否

6）产生该日志的活动的用户相关信息：无用户活动。

7）遵循的调查规程、使用的工具、屏幕截图等：上文描述的“初始调查”规程。

8）采取的调查行动：根据上一节描述的“初始调查”规程，确定该日志条目后面有一次成功的注销。确切地说，在同一天的不到一分钟之后，出现如图 16.12 的日志。

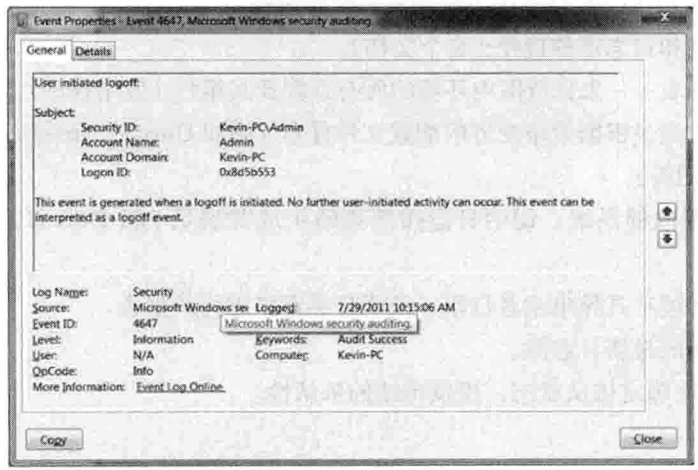


图 16.12　Windows 7 事件日志条目

这一条目说明，异常日志条目中所引用的操作成功注销。

9）日志分析过程中接触的人：无。

10）分析过程中确定的影响：影响被确定为“低”到“无”；没有对任何功能造成不利

影响，没有系统风险。

11）行动和缓解措施（如果必要）建议：不需要缓解措施，将这个日志条目加到基线中，未来将忽略它。

这种日志簿可以用作依从性证据，因为它建立了日志异常的跟踪，这是 PCI DSS 测试规程 10.6.a 中要求的："获取和检查安全策略和规程，验证它们包含每天至少审核一次安全日志的规程，异常情况的跟踪是必需的。"

如果必要，日志簿（不管电子还是纸质形式）可以提供给 QSA 或者其他审计人员（Anton Chuvakin 博士的博客，2011）。

16.9 PCI 依从性证据包

总体来说，创建一个"PCI 依从性证据包"以向 QSA 展示是很有用的，它将确立 PCI DSS 日志记录需求的 3 个关键部分：

- 日志记录的存在和充足性
- 日志审核
- 异常处理

可以在评估之前准备证据包，但是以持续的方式维护它要容易得多。例如，保留如下内容的纸质或者电子拷贝：

1）覆盖所有 PCI DSS 范围内系统的日志记录策略。

2）日志记录和日志审核规程（这个文档）。

3）日志源列表——来自范围内环境的所有系统及其组件（应用程序）。

4）表明日志记录根据策略配置的配置文件样本（例如 Unix 的 /etc/syslog.conf，Windows 审计策略屏幕截图等）。

5）来自范围内的系统，说明日志按照策略生成并满足 PCI DSS 日志记录需求的日志样本。

6）从日志管理工具导出或者打印、说明日志审核进行的报告。

7）上面定义的最新日志簿。

这样就能始终确定依从状态，提供持续的依从性。

16.10 管理报告

除了依从性证据之外，可以利用验证活动向高管报告日志管理计划、过程和规程的成功。

上一小节积累的作为 PCI DSS 依从性证据的数据也可以用于管理报告。具体地说，可以

生成如下报告：

- 日志记录的存在和充足性：
  - 这一部分没有实用的管理报告。
- 日志审核过程的存在及其实施：
  - 日志策略和规程变更。
  - 处于日志审核之中的应用程序。
  - 审核的日志条目。
- 异常处理过程及其实施：
  - 按照类型、分析人员姓名等列出的已处理日志异常。
  - 升级到事故响应的异常。
  - （如果合适的话）由于及时升级或者事故预防而降低的风险。
  - 由于及时升级或者事故预防而节约的资源。
  - 由于日志审核带来的应用程序性能改进。
- 其他日志管理计划报告：
  - 总体依从性准备（PCI DSS 和其他）。

最后，我们来总结组织应该执行的，与日志审核相关的所有运营任务。

16.11　定期运营任务

下面的小节包含与日志记录和日志审核相关的运营任务。有些任务在过去的章节中详细做了说明；其他任务则是成功实施 PCI DSS 日志审核计划所需的辅助任务。

16.11.1　每日任务

表 16.6 包含每日任务，负责执行它们的角色，以及它们的执行中创建的记录或者证据。

表 16.6　每日任务

任　务	责任方	证　据
按照每日日志审核规程的描述，审核前一天生成的各类日志	安全管理员、安全分析人员、（经过授权的）应用程序管理员	日志管理工具上运行的报告记录
（必要时）按照调查规程中的说明，调查异常日志条目	安全管理员、安全分析人员、（经过授权的）应用程序管理员	调查事件所记录的日志簿条目
（必要时）采取必要的行动缓解、补救或者协调调查的结果	安全管理员、安全分析人员、（经过授权的）应用程序管理员、其他各方	调查事件和采取的行动所记录的日志簿条目
验证所有范围内的应用程序都进行日志记录	应用程序管理员	创建一个数据表记录这些活动，供以后评估
（必要时）如果日志记录被禁用或者停止，则启用之	应用程序管理员	创建一个数据表记录这些活动，供以后评估

16.11.2 每周任务

表 16.7 包含每周任务，负责执行它们的角色，以及它们的执行中创建的记录或者证据。

表 16.7 每周任务

任 务	责任方	证 据
（如果得到 QSA 的批准）按照每天日志审核规程的描述，审核前一天非关键应用程序生成的各类日志	安全管理员、安全分析人员、（经过授权的）应用程序管理员	• 日志管理工具上运行的报告记录 • QSA 批准较低频率日志审核的记录，以及批准的理由
（必要时）按照调查规程中的说明，调查异常日志条目	安全管理员、安全分析人员、（经过授权的）应用程序管理员	调查事件所记录的日志簿条目
（必要时）采取必要的行动缓解、补救或者协调调查的结果	安全管理员、安全分析人员、（经过授权的）应用程序管理员、其他各方	调查事件和采取的行动所记录的日志簿条目

16.11.3 每月任务

表 16.8 包含每月任务，负责执行它们的角色，以及它们的执行中创建的记录或者证据。

表 16.8 每月任务

任 务	责任方	证 据
准备关于所调查的日志条目的报告	安全分析人员、安全经理	准备好的报告（存档）
有关观察到的消息类型的报告	安全分析人员、安全经理	准备好的报告（存档）
有关观察到的新消息类型的报告	安全分析人员、安全经理	准备好的报告（存档）
（如果得到 QSA 的批准）按照每天日志审核规程的描述，审核前一天在非关键应用程序上生成的各类日志	安全管理员、安全分析人员、（经过授权的）应用程序管理员	• 日志管理工具上运行的报告记录 • QSA 批准较低频率日志审核的记录，以及批准的理由
（必要时）按照调查规程中的说明，调查异常日志条目	安全管理员、安全分析人员、（经过授权的）应用程序管理员	调查事件所记录的日志簿条目
（必要时）采取必要的行动缓解、补救或者协调调查的结果	安全管理员、安全分析人员、（经过授权的）应用程序管理员、其他各方	调查事件和采取的行动所记录的日志簿条目

16.11.4 季度任务

表 16.9 包含了季度任务，负责执行它们的角色，以及它们的执行中创建的记录或者证据。

表 16.9 季度任务

任 务	责任方	证 据
验证所有 PCI 范围内的系统都记录日志，且审核了日志	安全分析人员、安全经理	
审核每日日志审核规程	安全分析人员、安全经理	
审核日志调查规程	安全分析人员、安全经理	
审核收集的依从性证据	安全分析人员、安全经理	
审核依从性证据收集规程	安全分析人员、安全经理	

16.11.5　年度任务

表 16.10 包含年度任务，负责执行它们的角色，以及它们的执行中创建的记录或者证据。

表 16.10　年度任务

任　务	责任方	证　据
审核日志记录和日志审核策略	CSO	
在 QSA 评估之前审核依从性证据		
对异常进行现场测试	按照需要	

16.12　其他资源

下面是 PCI DSS 日志审核计划和日志管理的有用参考：

SANS CAG/CSC. “Twenty Critical Security Controls for Effective Cyber Defense: Consensus Audit Guidelines”：

http://www.sans.org/critical-security-controls/.

审计日志的相关控制可以在如下地址中找到：

http://www.sans.org/critical-security-controls/control.php?id=6.

NIST 800-92 Logging Guide. “Guide to Computer Security Log Management: Recommendations of the National Institute of Standards and Technology by Karen Kent and Murugiah Souppaya”

http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf.

NIST 800-66 HIPAA Guide. “An Introductory Resource Guide for Implementing the Health Insurance Portability and Accountability Act (HIPAA) Security Rule”

http://csrc.nist.gov/publications/nistpubs/800-66-Rev1/SP-800-66-Revision1.pdf.

16.13　小结

日志审核是进入调查过程的起点，而且能够为你进行满足企业自身、监管或者其他强制要求的活动提供保证。本章的关键点可以总结为：

1）确保你从所有关键系统中捕捉日志数据。如果没有收集到任何内容，就无法执行日志审核和响应。

2）开发一个日志记录策略，描述收集和聚合日志之后保存日志数据（留存）的时长。

3）在日志审核过程中，建立常见模式的基线。

这一过程可以人工或者使用自动化工具完成，还要理解如何找出已知错误消息。

4）根据环境组成建立一个升级策略。

这取决于你的组织结构和职责部门。

参考文献

Chuvakin, A. (2011, March 28). Log review for incident response: Part 2. Prism Microsystems. Web June 10, 2011. <http://www.prismmicrosys.com/EventSourceNov2010.php>.

Chuvakin, A. (2011). PCI DSS and logging: What you need to know. N.p. Web October 4, 2011. <http://www.slideshare.net/anton_chuvakin/pci-dss-and-logging-what-you-need-to-kn>.

Chuvakin, A. (2011). http://www.securitywarriorconsulting.com/siemblog/.

Complete PCI DSS Log Review Procedures Part 8. Insert name of site in italics. Web October 4, 2011. <https://www.infosecisland.com/blogview/10796-Complete-PCI-DSS-Log-Review-Procedures-Part-8.html>.

Dr Anton Chuvakin Blog PERSONAL Blog: Complete PCI DSS log. <http://chuvakin.blogspot.com/2010/12/complete-pci-dss-log-review-procedures_07.html>.

Requirements and Security Assessment Procedures (2010, October). *Payment Card Industry (PCI) Data Security Standard*. PCI Security Standards Council. Web July 24, 2011. <https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf>.

Schmidt, K. J. (2007, April) Introduction to security log analysis. Community Bankers Association of Georgia. Georgia, Callaway Gardens. Presentation.

Williams, Branden R., & Chuvakin, Anton. In Tony Bradley (Ed.), (2007). *PCI compliance: Implementing effective PCI data security standards*. Syngress Publishing.

第 17 章 Chapter 17

对日志系统的攻击

本章要点

- 各类攻击
- 攻击案例研究

17.1 概述

健壮的日志分析系统依赖于所要分析的日志数据的完整性。该系统必须对修改和删除数据的企图有适应能力。在此基础上，它还必须对日志数据的访问进行细粒度的控制。如果日志数据作为法律上的证据，说明日志数据完整性的能力对于该数据是否被视为可接受的证据有着很大的影响。本章探索日志记录系统可能遭受的攻击，并对缓解或者检测这些攻击的解决方案提出建议。

为什么攻击者会以日志记录系统为目标？首先，我们在这一背景下所说的“攻击者”是什么含义？攻击者是进行未授权活动的人，不管他们是“外部人员”还是“内部人员”。

外部人员的典型例子之一是邪恶的互联网黑客。内部人员可能是你自己的系统管理员。攻击者可能不直接寻求入侵计算机，但是可能参与某些恶意活动，例如挪用公司的资金，或者向竞争对手出售商业机密。

17.2 各类攻击

攻击以日志为目标的原因多种多样。首先，聪明的攻击者通常希望避免被抓住。因为日

志数据提供了活动的证据，他希望避免这些信息被发现。更聪明的攻击者可能希望通过误导来隐藏他的痕迹，使观察者认为发生的是其他事件。除了隐藏痕迹之外，攻击者可能在日志中发现对自己有用的信息，例如事务数据，或者有助于攻击其他系统的信息，如密码或者账户编号。因此，攻击者有不被日志记录、销毁日志、修改日志的动机，也希望窥探来自各种系统的日志，以收集情报。

17.2.1 攻击什么

对日志记录基础设施的攻击可以在基础设施的任何一点进行：

- 源：日志消息生成的主机，攻击系统或者日志记录应用程序本身都是可行的选择。
- 传输：源和日志主机之间的网络，或者处理和存储之间的网络。
- 收集器或者收集日志的代理：收集来自各种源的日志的场所。
- 日志数据存储：存储日志信息的数据库系统（如果你为这一目的使用数据库）。也适用于存档日志数据的任何机制，包括利用训练有素的、以备份磁带为生的人。
- 分析：完成分析的系统和分析过程本身。
- 最后，攻击者可以攻击任何日志管理都无法避免的部分，查看数据和作出决策的分析人员。

攻击本身可以用计算机威胁分析的“CIA”模型分类：

- 机密性：攻击者读取日志数据的能力。
- 完整性：攻击者修改、破坏或者插入“构造的”日志数据的能力。
- 可用性：攻击者删除日志数据或者禁用日志记录系统的能力，或者拒绝分析功能操作的能力。

例如，远程入侵者使用的 rootkit 中常常发现一种简单的攻击，即删除 /var/log 中的任何文件。这种攻击可以通过使用加固的集中日志主机来缓解，在原始主机上的数据仍然会被删除，但是日志主机上还有可用的拷贝。对日志主机的这类攻击更为致命。

另一种简单的攻击是对源和日志主机之间的网络进行洪泛攻击，造成丢包，阻止日志数据到达日志主机。

本节探索这些攻击和其他一些攻击的细节。

17.2.2 对机密性的攻击

对日志数据的机密性攻击与隐藏痕迹的入侵者无关，而与收集情报相关。这里所说的情报可能是关于系统和网络的信息，这种信息可以用于其他攻击，信息的收集可能本身就是攻击。即使数据本身不是敏感的，法律或者公司政策可能将该数据的曝光视作违规。

入侵者能在你的日志中找到什么？下面是日志记录领域中可能有用的一些例子：

- 你正在运行的应用程序，以及运行它们的主机。日志往往显示应用程序版本，也可能包括有关应用程序配置的信息。所有这些数据可以用来寻找漏洞。
- 你在日志中可能注意到什么，这是为了确定如何避免被注意，或者为了不被记入日志

（原因同前）。

- 有用的信息片断，如谁有访问特权，人们何时登录等。
- 偶然曝光的证书。例如，用户在提示输入用户名时不小心输入了他们的密码，直到身份认证失败之后才发觉。错误输入的密码可能会被当作用户名记入日志中的“登录失败”消息。
- 可以窃取的数据或者事务记录的位置，这可能是攻击的真正目标。

1. 源机密性

得到日志数据的最简单方法是访问生成这些数据的地方——消息源。过高的权限或者间接访问可能使攻击者能够阅读日志数据。下面的简单例子中，日志文件可以被任何人阅读：

```
#ls -l /var/log/messages
-rw-r--r-- 1 root root 327533 Oct 9 15:25 /var/log/messages
```

如果非特权用户能够读取日志文件，他们就更加容易诊断自己的问题，但是他们可能阅读不应该阅读的内容——例如系统管理员在用户名（root）的位置输入了自己的密码。

间接访问的例子之一是在日志文件存储的磁盘分区所用的设备文件上为所有人设置可读权限：

```
# ls -l /dev/hd*
brw-r----- 1 root disk 3, 0 Oct 9 04:45 /dev/had
```

在上述例子中，“disk”组的任何用户都可以读取整个磁盘。在本例所处的特定系统中，该组的唯一成员是“root”。

```
# grep disk /etc/group
disk:x:6:root
```

这样的设置还不错。但是如果另一个用户能够成为该组的成员，他们就可以读取磁盘上的任何内容（提醒你一句，不仅是日志文件，而是任何内容！任何人的影子文件？）。

阅读原始磁盘数据听起来很困难，但是只要有标准的 Unix 工具和一点耐心就能完成。下面是一个例子：

```
dd if=/dev/hda2 bs=1M | strings | grep '^[A-Z][a-z][a-z] *[0-9][0-9]*
   [0-9][0-9]:[0-9][0-9]:[0-9][0-9] [^ ]'
```

dd 是一个简单的工具，用于在设备之间传输数据。参数指定输入设备和“块尺寸”（在这种情况下告诉 dd 一次读取多少数据）。strings 命令读取二进制输入，并打印任何看上去像可读字符串的内容（一系列可打印字符）。grep 用一个正则表达式匹配输入（更多细节参见第 8 章）。这里提供的模式与 syslog 消息开始处时间戳的匹配。输出如下所示：

```
Sep 5 07:29:00 linux syslogd 1.4.1: restart.
Sep 5 07:29:05 linux kernel: klogd 1.4.1, log source = /proc/kmsg
   started.
```

```
Sep 5 07:29:05 linux kernel: Inspecting /boot/System.map-2.6.4-52-
   default
Sep 5 07:29:05 linux kernel: Loaded 23439 symbols from /boot/System.
   map-2.6.4-52-default.
```

这正是在日志文件中看到的样子。

这个例子在大的磁盘分区上需要花一些时间，但是攻击者通常有的是时间。而且，更高级的工具可以在更短的时间内完成。

这种方法不完全准确，因为它可能发现任何类似日志消息的数据，包括删除的数据。在创建这个例子时，我们的测试脚本令人惊讶：

```
Feb 6 22:45:05 glen kernel: HiSax: Driver for Siemens chip set ISDN
   cards
Feb 6 22:45:05 glen kernel: HiSax: Version 2.1
Feb 6 22:45:05 glen kernel: HiSax: Revisions 1.15/1.10/1.10/1.30/1.8
Feb 6 22:45:05 glen kernel: HiSax: Total 1 card defined
Feb 6 22:45:05 glen kernel: HiSax: Card 1 Protocol EDSS1 Id=HiSax (0)
Feb 6 22:45:05 glen kernel: HiSax: AVM driver Rev. 1.6
```

显示的结果是 ISDN 配置教程中的日志消息示例，该教程随操作系统安装。

为了避免这类攻击，需要验证日志文件及目录的权限已经严格限制在特权用户上。还要确保日志所有上级目录只能由特权用户写入。否则，攻击者可以重命名目录和文件，并创建他可以读取的文件。

如果有需要访问日志的用户，创建一个单独的组（例如“logs”）并为该组提供读权限。还应该在磁盘和分区设备（例如 /dev/hde*）上检查权限。确保任何日志轮转程序在文件上设置了合适的权限。

除了对文件的直接访问之外，如果攻击者能够修改 syslog 配置文件，他就可以在另一个位置建立日志的第二拷贝，或者将日志转发到另一台主机。配置文件默认情况下一般不是所有人可读的，你应该保持这种配置。实际上，很少有理由将任何常规的文件设置为所有人可读。当然，具有根权限的攻击者仍然能够修改该文件，但是这样的攻击者可以做任何事情，所以这时你所要考虑的问题比日志文件要严重得多。

有些人建议将配置文件保存在另一个位置，这样日志所在位置就不会那么明显。而且，试图修改这些文件的自动化 rootkit 使用默认的位置，无法完成这种攻击。rootkit 是一组入侵工具包，它们往往设计为自动删除痕迹并安装后门。是否耗费精力去混淆日志文件的位置，完全取决于你。

2. 传输中的机密性

如果你打算将日志数据转发到一台集中日志主机，具有源和日志主机之间网络访问权限的攻击者可能够拦截传输中的数据。这类攻击可以在任何一个端点、利用对任何一台主

机上网络设备的访问权限或者通过两台主机之间任何网段上的“嗅探”（拦截流量）来完成。拦截网络流量并不像某些人想象的那么难。下面是用 tcpdump 嗅探 syslog 流量的一个简单例子：

```
# tcpdump -s 1600 -w syslog-dump.dmp src host 172.16.90.128 and proto
  UDP and port 514
```

上面这行命令收集来自 172.16.90.128 主机端口 514（syslog 端口）上的所有用户数据报协议（UDP）数据包，并将其写入文件 syslogdump.dmp。要查看结果，再次使用 tcpdump：

```
# tcpdump -r syslog-dump.dmp source -X -v
```

结果如下：

```
12:02:29.402842 anton.syslog > anotherbox.syslog: [udp sum ok] udp 36
  (ttl 128, id 47609, len 64)
0x0000 4500 0040 b9f9 0000 8011 5fdd ac10 5a80 E..@......_...Z.
0x0010 ac10 6e35 0202 0202 002c a5c3 3c31 3839 ..n5.......,..<189
0x0020 3e73 7973 6c6f 673a 206b 6c6f 6764 2073 >syslog:.klogd.s
0x0030 7461 7274 7570 2073 7563 6365 6564 6564 tartup.succeeded
```

这不像查看日志文件那么容易，但是对于攻击者来说已经足够。我们在本章中介绍的字符串还能使 tcpdump 数据更加可读。而且，plog 工具（在第 6 章中讨论过）将这些数据转换为 syslog 格式记录。

现在，有些人会说“我们采用交换式网络，攻击者无法在网络上嗅探流量……”交换机常常会泄露流量。有些低端交换机并没有真的在所有端口上采用交换，端口组可能共享流量。而且，有些交换机可以通过使其 MAC 表溢出而变为 Hub。

所以，不要依靠网络设备来保持数据机密性，它不是设计用于这一目的。

缓解日志数据的网络拦截问题有各种不同方法，它们都基于传输中日志数据的加密。syslog 可靠 RFC 提供日志数据流的加密，如果你使用支持该 RFC 的 syslog 客户端和服务器，就可以得到这一功能。我们在第 3 章中讨论了该 RFC 和客户端。syslog-ng 允许用 PGP 或者 crypt 加密日志数据，也可以使用你喜欢的其他加密方法（Rot13 总是个办法）。

这种方法在第 5 章中关于 syslog-ng 的小节里做了介绍。如果你使用基于 TCP 的日志传输，流量还可以用 sslwra、stunnel 通过 SSL 隧道传输，或者使用一个 SSH 隧道。

注意，控制其中一个端口（源或者日志主机）的攻击者可以在日志数据加密前拦截，或者拦截加密密钥。使用公钥加密方法（如我们介绍的 PGP 示例），可以不将私钥保存在源或者日志主机（私钥只需要在加密数据时使用，而不需要出现在解密的位置），从而缓解后一种攻击。

当然，如果你相信自己很好地控制了客户端和日志主机间的网络，可以选择不担心日志流量的拦截问题，这取决于你自己的判断和风险分析。

3. 日志主机的机密性

保护源上的数据机密性所涉及的所有问题都适用于日志主机。此外，由于日志主机一般

只用于收集和检查日志数据，你可以将日志主机上的账户限制在需要访问日志数据或者管理日志主机的账户。

对于某些环境，隐藏的日志主机可能更为必要。隐藏的日志主机在网络上不可见，但是网络上的主机仍然可以向其发送日志，详见第 6 章。

4. 日志存储的机密性

存储日志的数据库是机密性攻击的另一个向量。保护日志主机的相同问题也适用。此外，具有数据库用户名和密码的攻击者可能连接到数据库并查询数据，或者拦截日志主机和数据库服务器之间传输的事务。

数据库服务器应该被看作和日志主机一样关键，并以相同的方式保护。数据库连接应该限制在日志主机和用于日志分析的系统上。这些连接应该加密，以避免拦截。数据库用户的访问规则应该加以限制，使得只有授权用户能够查询数据。

将日志放入数据库中有许多方法。所有数据库都支持直接网络连接，使用不同的身份认证级别。在数据库中存放日志的另一种方法是使用文件传输机制，从日志主机将数据“拉”到数据库服务器，然后从本地文件中将数据加载到数据库。

大部分数据库都提供数据库访问的某些控制方法，可以通过主机和用户限制。mysql 使用 GRANT 语句，为两种限制提供相同的机制：

```
mysql> GRANT ALL PRIVILEGES ON db.* TO
   someuser@'10.1.2.0/255.255.255.0';
```

上述语句允许用户“ someuser”从 10.1.2.0 网络上的任何主机连接到任何数据库，进行任何操作。你可能希望进一步限制权限，将修改的能力限制在日志主机上的用户，对其他人只允许读权限：

```
mysql> GRANT INSERT, UPDATE PRIVILEGES ON logdb TO
logwriter@loghost.example.com;
mysql> GRANT SELECT PRIVILEGES ON logdb TO logreader@analsysis-station.
   example.com;
```

上述例子允许用户 logwriter 写入数据库，只能从日志主机上进行。另一个用户 logreader 只允许查询数据库，但是不能修改，只能从日志分析工作站上操作。注意，logwriter 用户不能查询数据，只能插入和修改数据。

Postgresql 有类似的访问控制，但是实现的方式不同。主机和用户的访问控制被放入文本文件 pg_hba.conf。为了实现 mysql 示例中的相同访问控制，我们在 pg_hba.conf 中输入如下配置：

```
hostssl logdb logwriter 192.168.3.2/32 password
hostssl logdb logreader 192.168.3.5/32 password
```

这里的 192.168.3.2 是日志主机的 IP 地址，192.168.3.5 是分析系统的地址。还要注意，

我们选择了一个访问方法，使用 SSL 加密数据流。然后，SQL GRANT 命令用于控制每个用户对数据库能进行何种操作：

```
admin=> GRANT INSERT, UPDATE logdb TO logwriter;
admin=> GRANT SELECT logdb TO logreader;
```

Oracle 用另一种机制提供类似的访问权限。

“拉”方法的好处是数据库不一定要允许任何网络连接，消除了数据库的一个攻击向量。

正如我们已经提到的，如果你不打算管理数据库访问控制，可以将数据拉取到数据库服务器，从本地加载。

例如，可以使用“rsync”命令，将 SSH 作为传输手段，如：

```
# export RSYNC_RSH = /usr/bin/ssh
# rsync -avz logwriter@loghost.example.com:/var/log/* /var/log/store/
  remotehost1
```

这将拷贝日志主机 /var/log 目录中之前尚未复制的所有文件。你仍然必须决定是否允许分析工作站上的读权限，但是至少不需要授予通过网络写入的权限。有些人可能认为“拉”方法较为安全，因为攻击者无法直接通过网络攻击数据库服务器。

5. 分析的机密性

分析系统需要日志数据的访问权限，因此也是获取日志数据访问权限以及日志分析方法相关信息的攻击点。和数据库及日志主机一样，分析工作站应该通过将访问权限限制在使用工具的人、验证文件权限和加密流量，加以保护。

当然，攻击者可以在安全分析人员查看日志数据时“肩窥”，这是另一种机密性攻击。对此我们没有技术上的解决方案。

17.2.3　对完整性的攻击

对日志完整性的攻击是通过重写或者插入虚假数据、删除数据，破坏真实数据的能力。攻击者破坏数据通常是为了掩盖活动的证据。最常见的方法是删除相关的数据。更聪明的攻击者可能只删除表示其在系统上的活动的日志消息，或者简单地修改消息，使得它们看上去像是合法的，因而不会引起注意。内部攻击者甚至可能想要修改日志，以“陷害”其他人，或者让人觉得其他人犯了错误。

上述的几类攻击者都想通过插入表示其他活动发生的虚假消息，分散人们的注意力，使他们忽视攻击者的活动。例如，攻击者可能插入虚假的错误消息，使系统管理员忙于跟踪根本不存在的问题。另一种技术是显示同一网络上的另一个主机发起的虚假攻击。

如果日志数据用于提供证据，能够断定日志数据的完整性，对于日志是否被接受为证据可能有影响。攻击者如果可以说明日志不能免遭破坏，就可以声称日志是伪造的，不能作为其不法行为的证据。

1. 源完整性

和我们前面讨论的机密性问题类似，不合适的权限可能使攻击者在日志数据存储时直接修改。

例如，假定系统上的用户“anton”正在做坏事，不希望这些活动和他联系起来。如果anton能够修改存储的日志文件，他可以用“marcus”替代日志中所有“anton”，将对该活动的职责转嫁给他人。这很容易做到：

```
% sed 's/anton/marcus/g' < /var/log/logfile > /var/log/logfile.x; /bin/
  mv /var/log/logfile.x /var/log/logfile
```

（还有一个细节，syslog守护进程必须发送一个SIGHUP信号，才能开始写入到新日志文件）。

可以采用和机密性中我们所讨论的相同方法验证日志数据、设备、配置文件等。

此外，将日志转发到一台集中日志主机虽然不能避免这种攻击，但是创建了另一组日志，攻击者必须修改它才能有效地隐藏自己的痕迹。多组日志和多个日志信息来源更容易检测到攻击者的活动，即使某些日志文件遭到修改也仍然如此。当受到指控的攻击者生成“那不是我干的，是别人干的！”的日志时，多个数据源对防抵赖也很有用。“黑客”被发现时往往试图提出其他人进行攻击的证据。来自多个源的日志有时候能够对此提出证明。不仅在Unix上，Windows系统上也可能进行类似的攻击（修改或者删除日志），其他平台也会受影响。

对于事件日志，攻击稍难一些，因为文件是二进制格式，需要一个工具来读取和写入文件，这只需要少量编程。这种文件的格式有足够的文档，编写这类工具非常简单。顺便说一句，较新的Windows版本（如Windows 2008和Windows 7）使得这种攻击更加困难。

在支持扩展属性的Unix系统之上，日志文件可以设置为“仅附加”，避免普通用户覆盖该文件。当然，具有根权限的用户可以重置这个属性（参见关于具有根权限的入侵者的注释）。此外，仅附加文件阻止常规用户修改数据，甚至阻止具备文件写入权限的用户。而且，连根用户也需要额外的步骤，才能使文件再次变成可写入，这能阻止文件的“无意”覆盖。

生成虚假日志消息是难以缓解的问题。创建消息相当简单，例如，人们可以轻松地使用logger程序。

下面是来自Linux系统的一条真实的syslog消息：

```
Nov 11 22:41:00 ns1 named[765]: sysquery: findns error (NXDOMAIN) on
  ns2.example.edu?
```

使用Unix和Linux系统上的logger标准工具，创建一条匹配原始消息的虚假消息：

```
$ logger -t "named[765]" "sysquery: findns error (NXDOMAIN) on ns2.dac.
  net?"
```

生成的虚假消息如下：

```
Nov 11 22:45:51 ns1 named[765]: sysquery: findns error (NXDOMAIN) on
  ns2.dac.net?
```

由于syslog机制的特性，很难抵御这类进攻。可以从系统删除日志记录器程序，但是攻

击者可以轻松地引入自己的拷贝。

实际上，可以创建一个受限的日志记录器程序。这包括更改日志记录套接字（通常是 /dev/log）的权限，删除所有人都可以写入的权限。当然，这样会有一些不是以根用户运行的应用程序无法写入日志设备。可以创建一个用户组来访问日志设备，例如，创建 log 组，将日志设备组的属主改为 log，这些应用程序以守护进程的身份运行于一个角色账户之下[⊖]，该账户可以添加到 log 组。但是，我们没有实际尝试过。

但是，由常规用户运行的应用程序无法记录日志。可以为日志记录器创建一个包装器，由其运行 setgid 日志，并在日志消息中插入真实的进程名（可能还有用户名）。

最后，对日志定时的攻击可以看作对完整性的一种特殊而具破坏性的攻击。

2. 传输中的完整性

标准 Unix syslog 消息通过 UDP 传输。UDP 数据包报头中的 IP 地址没有认证，所以没有办法知道它是否来自所声称的主机。而且，syslog 传输是单向的；服务器不向客户端发送任何确认。由于服务器接收的数据包被当成来自显而易见的发送者接受，具备网络访问权的攻击者可以注入看上去是从网络上合法主机发送而来的伪造 syslog 消息。

有许多程序可以生成任意的网络数据包，大部分基于 libnet。packit 是这种工具的一个例子，注意，packit 本身不是 syslog 生成工具，它的作者将其描述为“网络审计工具”。下面是使用 pckit 生成一条虚假 syslog 消息的例子：

```
packit -t UDP -d 172.16.110.53 -s 172.16.90.128 -D 514 -S 514 -vvv -p
   '0x 3c 31 38 39 3e 73 79 73 6c 6f 67 3a 20 6b 6c 6f 67 64 20 73 74
   61 72 74 75 70 20 73 75 63 63 65 65 64 65 64'
```

上述命令创建一个 UDP 数据包，发送给 IP 地址 172.16.110.53 的 514 端口，似乎来自于 172.16.90.128，内容为“ syslog:klogd startup succeeded”（你在例子中看到的是表示消息串的 16 进制代码。这条消息用于使系统管理员认为系统突然重启。）

具有源和目标之间网段访问权限的攻击者可以用 ARP 欺骗中间人攻击拦截和修改 IP 流量（劫持会话）。对这类基于网络的攻击，最好的防御措施是使用验证消息完整性的机制。syslog-sign 协议提供了这种机制。另一个选项是使用安全套接字层（SSL）甚至 Internet 协议安全（IPSEC）。

如果这些消息完整性机制在你的网络上不可行，你可以在网络边界实施合适的防欺骗过滤器[⊜]，并限制日志主机接受的消息来源主机，以缓解这类攻击。

ARP 欺骗可以通过强制日志主机在每个日志源上有一个条目（如果你希望考虑得周全一些，反之亦然）来避免。这可以用 Unix 上的 arp 指令完成：

⊖ 角色账户是专为运行特定守护进程而创建的账户

⊜ 这是你应该采取的措施，我们建议同时使用入站和出站过滤器，使处于一个网段上的攻击者无法伪装成在另一个网段上或者来自外部主机

```
# arp -s loghost.example.com AA:BB:CC:DD:EE:FF
```

上述命令为 loghost.example.com 创建一个 arp 表条目，MAC 地址为 aa:bb:cc:dd:ee:ff。你可以用如下命令查看 arp 缓存状态：

```
# arp -a" fw.example.com (10.0.0.1) at 00:10:4B:26:46:A0 [ether] on
   eth0 loghost.example.com (10.0.0.127) at AA:BB:CC:DD:EE:FF [ether]
   PERM on eth0
```

第二行的“PERM”表示该条目已经永久性地加入到 arp 缓存，不会被覆盖。

当然，人工维护 arp 缓存很麻烦，移动主机时会导致混乱。此外，当源和日志主机在不同网段上时，人们必须维护网关而非日志主机的静态条目，这会带来更大的复杂性和潜在的混乱。

有些人通过串口线连接源和日志主机，避开网络拦截和修改的问题。远程攻击稍微困难一些，但是对于有多个位置的网站来说代价沉重。

3. 日志主机的完整性

对日志主机的完整性攻击基本上和源上的攻击一样，适用相同的保护技术。但是，这种攻击的影响比对单一源的攻击更大，因为日志主机收集的所有数据的完整性都受到威胁。

4. 数据库完整性

和数据库上的机密性攻击一样，适用类似的规则。限制数据库用户写入或者附加到日志数据空间的能力。验证网络访问保护认证证书，并且能够避免会话劫持。最佳的方法是除了访问规则之外，还使用一个验证数据完整性的机制。

数据库给关键数据保护带来了两个挑战，“变动数据”和“静止数据”的处理。前者指的是到达数据库的数据，后者指的是数据库存储的数据。数据库安全是一个巨大的知识领域，我们只简单地介绍一些和保护日志数据库数据相关的挑战。

变动数据通常通过加密从应用程序到数据库的连接，或者使用“推”日志集的方法（本章前面已经提到过）来保护。

加密连接的主要挑战在于，没有 HTTPS（HTTP+SSL）这样的标准化机制。现在有通过 SSL 传输 SQL 命令和数据的“SQLS”，但是没有广泛应用。

例如，最新版本的 MySQL 数据库（www.mysql.com）支持原生的 SSL 加密。在 Windows 领域，Microsoft SQL Server（MSSQL）也可以用 SSL 加密连接。Oracle 8 和 9 的 SSL 支持需要“Oracle 高级安全性”(OAS)。

保护存储在数据库中的数据的唯一可靠方法是数据库加密。在本书编著的时候，很少有数据库原生支持加密，需要使用第三方软件。

5. 分析的完整性

对分析完整性的攻击试图动摇对分析系统准确度的信心。假阳性太多，系统就没有用

处。假阳性削弱了对该系统捕捉任何信息的信心。下面是对分析系统完整性攻击的三个位置：

- 处理的数据（输入）
- 用于分析的工具（系统）
- 结果的显示（输出）

通过修改输入数据，攻击者可以导致系统报告虚假的攻击，或者不报告实际的攻击。分析系统用于访问数据的方法应该有完整性保护机制。如果这种方法通过数据库，则适用于存储访问的相同规则。如果通过文件服务器，应该加固传输安全，避免会话劫持和数据流破坏。

以分析系统为目标，攻击者可以更改分析工具，使系统不能像预期那样表现。分析系统的权限应该限制为仅供授权用户使用，用于分析的软件也应该有合适的修改保护。考虑对使用软件和修改软件的用户进行权限分离。应该采用修订控制或者用其他软件跟踪对软件的变更（包括配置文件），并定期验证软件的完整性。

最后，用于显示数据的系统可能遭到篡改，以更改或者抑制显示给分析人员的数据。这可以通过篡改系统创建的报告、创建虚假报告或者在数据审核中劫持会话来实现。

输出数据的安全应该和输入数据的安全同样看待，限制有写权限和系统访问权限的人。

显示系统也应该对会话劫持免疫。例如，如果使用一个 Web 服务器审核报告，传输应该使用 SSL 和合适的签名证书，以及密码控制访问。

17.2.4　对可用性的攻击

对可用性的攻击旨在阻止合法用户访问数据和系统。如前所述，攻击者的常用技术之一是获得系统访问权限，删除日志文件。有些可用性攻击和完整性攻击非常类似，使用相同的攻击向量。其他攻击则采用不同的方法，例如，导致系统崩溃，以阻止访问。

对可用性的攻击可以视为一种拒绝服务攻击。在这种情况下，“服务”意味着人力资源以及计算机资源。人员的时间往往是最宝贵的，时间的损失和网络断开同样是毁灭性的。

1. 源的可用性

最简单的可用性攻击是删除日志来源上的数据。

例如，在本书作者之一运行的作为 Honeynet Research Alliance（http://www.honeynet.org/alliance）一部分的 Linux 蜜罐中，捕捉了一个入侵会话。这是攻击者获得根权限之后发出的命令[⊖]：

```
# killall syslog
# rm ~/.bash_history
# ln -s ~/.bash_history /dev/null
```

⊖ 满足一下读者的好奇心，这个会话示例是入侵者通过 OpenSSL 攻击入侵受害系统，获取访问权限的结果。然后，攻击者利用对“ptrace”的攻击将他的权限提升到“root”

注意，第一条命令是杀死 syslog 守护进程，说明他希望阻止可能的远程日志记录（此外还阻止所有本地日志记录）。他删除根用户的 .bash_history 文件，销毁他所输入的任何命令的记录，并将其符号链接到 /dev/null，避免进一步记录任何新命令。然后，入侵者安装一个 rootkit，运行如下命令：

```
# zap /var/log/messages
```

上述命令用 0 覆盖 /var/log/messages 中的相关数据。和删除相比，数据置零以覆盖磁盘上存储的数据，会使其无法恢复。简单地删除文件会在磁盘上留下数据，可以用合适的工具恢复，这一点我们在对机密性的攻击中已经介绍过。[⊖]

许多黑客工具被用来“净化”登录记录，这意味着秘密地删除不想要的隐含记录。这类工具的常见例子有 zap、clear 和 cloak。这些工具以两种不同的方式运作：它们可能置零 / 替换二进制日志记录（从而用 0 记录填充文件，这很可疑）或者删除记录（从而使日志文件变短，这也很可疑）。两种方法都有缺点，都可以检测到。

二进制格式文件（如 wtmp 和 pact）也是删除的候选者。正如本书作者之一在他的前一本著作中所介绍的，有许多“净化”二进制文件的工具，可以用于 Unix 和 Windows 上。

下面是 Windows XP 日志被删除或者清除时发生的情况。假定攻击者试图从标准的 Windows Explorer shell 程序上删除 C:\WINDOWS\system32\config 中的 AppEvent.Evt，SecEvent.Evt 和 SysEvent.Evt 文件。

这一操作不能成功（见图 17.1），因为系统保护在用的文件。可以使用事件查看器应用程序清除这些日志（见图 17.2）。但是，这样安全日志不会被清除“干净”，将生成一个事件表明日志已经清除（见图 17.3）。

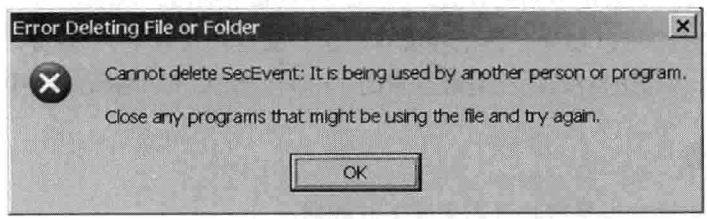


图 17.1　Windows 日志删除错误

事件日志也可以通过启动到另一个操作系统实例（例如用一张 Linux 软盘启动）或者使用特殊的日志清除工具来删除。WinZapper（http://www.ntsecurity.nu/toolbox/winzapper/）和 ClearLogs（http://www.ntsecurity.nu/toolbox/clearlogs/）是两个稍有点老，但仍然有效的工具。此外，日志记录可以通过违规使用轮转方案强制清除出该文件。

⊖ 在作者之一领导的对被入侵系统的调查中，使用标准的 Linux 文件系统工具和取证工具 TCT（Dan Farmer 和 Wietse Venema 开发的 Conroner Toolkit，详见 www.fish.com/forensics or www.porcupine.org/forensics）恢复日志文件。调查的详细情况可以在《Security Warrior》(O’Reilly，2004）一书第 20 章中看到。

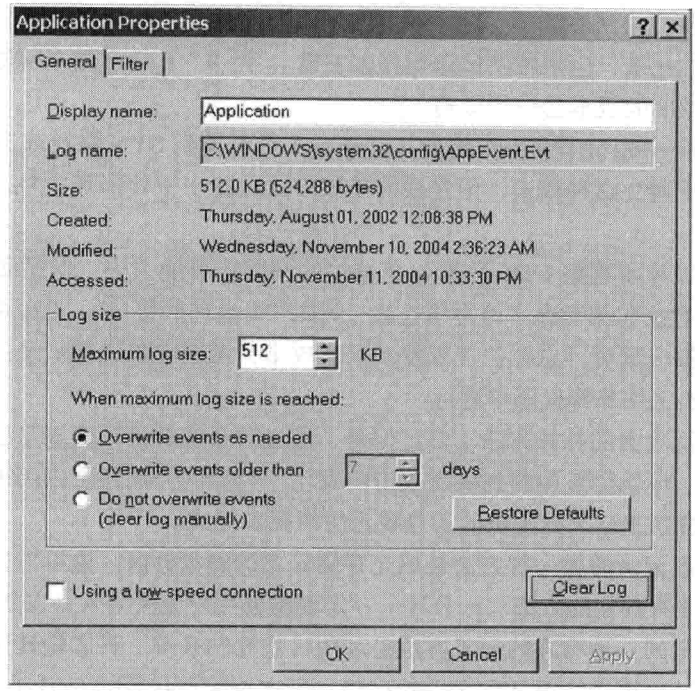


图 17.2　Windows 日志删除成功

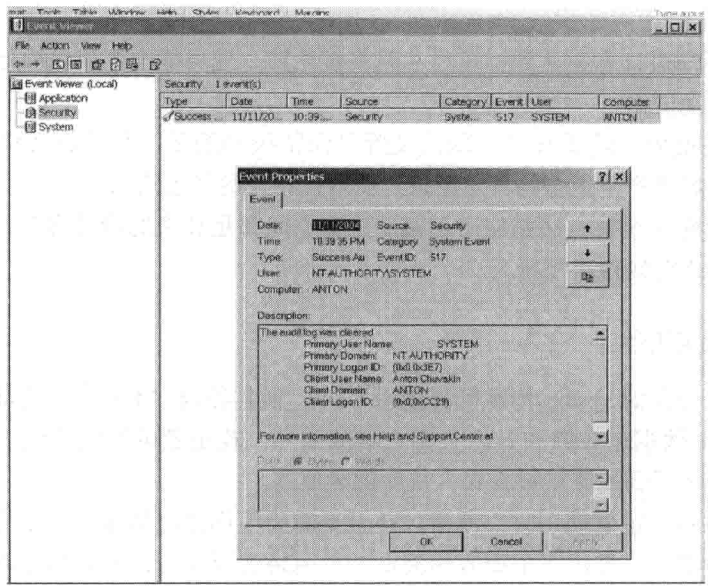


图 17.3　Windows 日志删除事件

涉及数据删除的攻击的缓解方法和完整性攻击的防范方法一样。

正如我们的例子中所看到的，除了删除本地日志文件之外，可以杀死日志守护进程，有效地阻止所有日志记录，包括转发日志到日志主机。“标准”syslog 的当前版本对拒绝服务攻击不敏感，但是较旧的版本会受到影响。[⊖]

基于 TCP 的 syslog 守护进程（例如 syslog-ng）可能遭到 SYN 洪泛攻击。虽然有一些试验性的技术有助于避免这种攻击，但是最简单的方法是除了所处网络之外，阻止对日志守护进程的网络访问。

此外，虽然大家都理解 UDP 数据包在繁忙的网络中可能丢失，但是我们注意到一个有趣的现象[⊜]：这些数据包甚至在本地网络连接（例如“本地环回接口”）上也可能丢失。攻击者可以简单地用虚假流量对日志守护进程发动洪泛攻击，抑制守护进程记录的消息。这样的攻击甚至可以在主机上用非特权用户进行。

对于本地洪泛攻击的补救方法不多。但是，通过对系统的活动建立基线，并搜索日志消息的不寻常数量，你就能够检测到洪泛攻击的发生。类似的方法还可以用来检测崩溃的日志服务器，这通过搜索来自某台主机的消息缺乏的情况来完成。

抑制本地日志记录的另一种方法是填充存储日志的磁盘分区。这可以通过填充日志来完成，攻击者也可能有其他的方法写入分区。一定要验证用户无法写入日志所在分区。

销毁日志数据的一种有趣的间接方法是利用日志轮转系统。日志轮转用于保护日志记录服务器免遭溢出，也用于组织日志文件的存档。日志轮转意味着储存旧日志文件（可能压缩或者移到另一个位置），并将日志记录切换到一个新文件。这在 Unix 和 Windows 系统上都能完成。

如果攻击者知道日志在达到特定尺寸时将被轮转，他可能会生成足以导致轮转的日志消息，删除他的活动证据。

Windows 根据大小轮转日志，默认设置只保留很小的量，在日志文件满时覆盖旧条目（见图 17.2）。很容易产生更多的日志记录，以取代相关的日志消息。

违规使用轮转的缓解方法是使用足够的磁盘空间和足够大的轮转容量，这就可以在攻击者填满日志文件之前发现违规行为。

2. 传输中的可用性

如前所述，标准的 syslog 传输机制使用 UDP，通常称为“不可靠”传输。网络上的高流量可能造成 UDP 数据包在源和日志主机之间被丢弃。攻击者可以用流量对网络进行洪泛攻击，使他不希望你看到的流量被丢弃，从而隐藏痕迹。

洪泛攻击并不一定要使用日志流量，甚至不需要使用针对网络真实主机的流量。攻击者躲避跟踪的好方法之一是用完全虚假的流量进行网络洪泛攻击，其目标是现有主机上未用的

⊖ 例如，“Cisco IOS Syslog 拒绝服务漏洞”（http://www.ciac.org/ciac/bulletins/j-023.shtml）可能导致 syslog 守护进程崩溃。

⊜ 我们认为自己发现了这一现象，但是没有足够精确的证据来确定。

端口，或者网络上不存在的主机，甚至另一个网络上的主机。这些流量在主机上不会显示为连接，也不会以任何其他明显的方式出现，只会在网络设备上留下利用率的统计数字。

“Ping 洪泛”是网络洪泛攻击的最简单方式之一，它很容易被忽略。Ping 程序的很多版本中有一个选项（通常是“-f”），告诉程序以至少每秒 100 个数据包（最高为主机能够响应的速度）的速度发送数据包。对与日志主机相同网段的一个主机发动 Ping 洪泛攻击，可能会导致发送到日志主机的数据包被丢弃，在日志主机上不会有出现问题的指示。实际上，唯一不正常的指示是路由器或者交换机上的流量计数器会显示丢包和大量的互联网控制消息协议（ICMP）流量。

下面是对日志主机运行的 ping 命令的示例：

```
# ping -f loghost
```

注意，Unix 系统上只有根用户才能使用 Ping 的这个选项。下面是非根用户在 Linux 系统上试图运行该命令时所发生的情况：

```
# ping -f loghost
ping: Operation not permitted
```

网络洪泛难以缓解。如何区分合法的流量和虚假的流量？如何知道网络流量的激增不是因为业务真的转好？你只能监控流量统计，搜索“不寻常”的事件（参见第 11 章）。

3. 日志主机可用性

和网络洪泛一样，日志主机也可能遭到洪泛攻击（利用虚假日志数据，或者就是虚假的流量）。用虚假日志流量对日志主机进行洪泛攻击有两种效果。首先，它可能造成合法的流量被网络或者日志主机的内核丢弃。此外，日志数据可能填充日志文件分区，迫使日志主机停止记录任何数据。（有时候这会造成主机系统故障）。

下面是一个 shell 脚本示例，可以用于从向日志主机转发日志的客户端机器上发起洪泛攻击：

```
i=0
while: do
 i=$i+1
 logger -p "Flooding $i"
done
```

递增变量的用处是避免 syslog 守护进程汇集“最后一条消息重复 X 次”的消息。

对你来说洪泛攻击不够快？那就在后台同时运行多个脚本。

这个例子对于本地日志和远程日志的洪泛攻击都有用。

基于 TCP 的 syslog 守护进程可以避免丢包，但是可能遭受 SYN 洪泛攻击，这种攻击可能使服务器崩溃，至少造成其拒绝接受更多连接，就像人们可以阻塞所有其他基于 TCP 的服务一样。与保护日志主机免遭完整性攻击的相同措施也适用于这种情况。

除此之外，日志主机和日志源一样容易遭到拒绝服务攻击，适用相同的保护性措施。

最后，数据库和日志主机一样容易遭到类似的攻击——网络洪泛、SYN 洪泛、使数据库崩溃的攻击，以及存储文件的删除。这里适用对完整攻击的相同建议。

4. 分析的可用性

除了禁用分析系统之外，用分析数据淹没系统，可能使其陷入困境。有效的数据精简技术有助于缓解这种攻击。

对可用性进行的一种攻击很有趣且常常被忽视，这就是对分析人员可用性的攻击。攻击者可能简单地依靠执行足够的不同攻击，或者注入数据使其看起来像攻击，使安全分析人员不知所措，或者使其花费时间去注意看起来更严重的其他活动，从而避开他们的视线。

而且，一个有太多“假警报”的分析系统很快会被认为是无用的并被放弃。这种攻击的最有名例子是警报洪泛攻击工具，如 stick、sneeze 和 snot，这些工具会造成早期入侵检测系统的问题。除了给运行入侵检测软件的系统造成压力之外，警报洪泛攻击工具生成许多警报，使监控系统的人员茫然不知所措。

日志事件的收集和分析必须以某种方式组织，最大程度地降低外部人员注入伪造数据的能力。通常，加密有助于解决这个问题。至少，它能够解决从源到收集点的传输中的数据伪造问题。例如，用通过 SSL、SSH 或者 IPsec 隧道的可靠 syslog 代替基于 UDP 的 syslog，能够阻止攻击者在收集点注入数据。

此外，某些机制还支持源身份认证，能够阻止攻击者引入虚假的日志记录来源，用它对系统进行洪泛攻击，压垮技术组件和分析人员。

攻击案例研究

在本节中，我们将描述一个虚构的公司在服务器入侵期间如何应对，在这一入侵之后紧接着发生了对日志记录基础设施的攻击。

“Rats R Us”是一家全球领先的企业和消费者端到端鼠管理公司。它提供了广泛的鼠管理产品和服务，包括鼠害预防、灭鼠、鼠相关事故清理以及为各种目的饲养老鼠、训练老鼠，以及为商业客户和政府培养定制的转基因老鼠。它通过网站、纸质目录和代理商及合作伙伴网络（组合于“RatOne”合作伙伴计划中）直接提供服务和产品。

该公司的中心办公室有数百名员工，包括所有 IT 人员。其余人员分布在区域销售机构和老鼠培育机构。数据中心也设立在中心办公室的同一建筑物中。

该公司的网站是公司收入的重要来源，所以“Rats R Us”投资了 IT 安全资源。“非军事区”（DMZ）架构有两个防火墙，保护公开的服务器（Web、电子邮件等），并将其与内部系统和互联网分离。基于 Web 的鼠交易系统的核心是一台电子商务应用程序服务器，连接到 Web 服务器，并用另一台防火墙分离。在外部防火墙之后布设了一个入侵预防系统，

防止对 DMZ 资产的攻击。

2009 年 5 月一个阳光明媚的日子，入侵预防系统开始生成关于公司网络上使用的 IRC（互联网中继聊天）程序的巨量警报。IDS 的配置是发现时不自动阻止这类活动，因为 IRC 通信可能是合法的活动。管理员查看每日活动报告，注意到日志中记录的所有活动中，90% 和全球多个 IRC 的服务器通信有关。经过严密的思考，他认为有某个系统遭到入侵，这个系统应该在 DMZ 中，因为内部系统没有到互联网的直接链接。通过查看 IRC 消息源，管理员定位 DMZ 中的一个可疑 Unix 系统——一台邮件服务器。

因为这台机器明显遭到入侵，管理员决定将其离线，以备调查和恢复。它立即从网络断开，但是保持电源开启。[⊖] 当事故响应团队查看该系统，他们注意到有一个未知的进程以根特权运行，这是入侵的明显征兆（例见《Issues Discovering Compromised Machines》），所以他们知道主机被入侵，但是不知道是如何入侵的。

响应团队尝试查看主机上的日志，却发现它们已经被删除。日志文件仍在，但是前 7 天没有记录任何内容。旧的日志仍然存在（由日志轮转脚本轮转并压缩）。旧日志没有提供任何入侵的情况。该服务器的配置是远程日志记录，所以该团队查看日志主机上的日志。

响应团队震惊地发现，syslog 服务器系统没有同一时期（7 天）来自任何服务器的日志。实际上，他们发现日志主机的 syslog 守护进程甚至没有运行。所以，不仅 DMZ 中的主机遭到入侵，集中日志主机即使没有被入侵，至少也遭到了 DoS 攻击。

网络入侵检测系统监视外部系统和 DMZ 机器之间的通信，但是没有发现任何攻击。网络 IDS（NIDS）上的日志包含正常的噪声，但是没有攻击活动。不幸的是，NIDS 不关注内部通信。

该团队试图查看防火墙上的外部连接日志。他们发现公司在防火墙上没有配置激进的日志记录，没有记录允许的连接，因为允许的连接被自动视为合法活动，因此没有必要记录日志。实际上，现实往往相反：被拦截的流量只说明防火墙正在工作，接受的流量反而是攻击的向量。

所以，团队做的下一件事是尝试恢复被删除的日志。他们使用昂贵的取证工具，花费几个小时制作硬盘映像，搜索日志数据。他们最终发现，日志不仅被删除，而且用 0 覆盖了。

没有太多证据可用，该团队只得假设入侵者使用了“0day 攻击”，获得对日志服务器的访问。这种攻击在日志文件中会留下痕迹，所以入侵者删除日志，避免留下攻击情况的线索。但是所有这一切都只是猜测，缺乏真实的证据。

他们重建日志主机，验证它已经打上了完整的补丁。提高日志记录级别，在日志主机上部署了“入侵预防系统”，希望它能阻止新的攻击。但是，对于日志主机是如何被入侵的没有太多信息，Rats R Us 仍然可能遭到同样的攻击。

⊖ 对于遭到入侵或者感染的系统，这个决定总是很困难：开机还是关机？两种方法都有道理。

该公司如何更好地保护日志主机，或者至少保留日志信息？他们可以将日志数据存档到某种一次性写入媒体或者仅附加文件，这不能避免入侵，但是提高了保留入侵时日志数据的机会。

日志主机可以配置为隐藏主机，只能从其控制台登录，这样远程攻击即便有可能进行，也很困难[⊖]。当然，这使得日志主机的管理变得更困难，但是如果日志数据有足够的价值，就值得这么做。

如果不采用隐藏主机，可以使用不同于网络上其他主机的操作系统，希望异构系统能够抵御在其他主机上有效的攻击。

这些解决方案各有其优劣。在你的环境中必须尽你所能。对明显的环节进行保护（如检查文件权限）通常相当简单。其他的事情要在你有相应的资源，并且符合网站需求的情况下进行。

17.3 小结

本章为理解针对日志记录系统的常见攻击打下了一个很好的基础。我们还提供了机密性、完整性和其他适用于日志系统的概念的例子。

参考文献

For brief intro on SYN flooding, see this old CERT advisory. <http://www.cert.org/advisories/CA-1996-21.html>.

libnet. <http://www.packetfactory.net/libnet/>.

Packit. <http://packit.sourceforge.net>.

Peikari, Cyrus, and Anton Chuvakin. *Security Warrior.* Beijing: O'Reilly & Associates, 2004. Print.

<http://www.eurocompton.net/stick/projects8.html>,<http://www.securiteam.com/tools/5DP0T0AB5G.html>, <http://www.securityfocus.com/tools/113> (see this article "Issues Discovering Compromised Machines" at <http://www.securityfocus.com/infocus/1808> for reference for compromise detection hints).

⊖ 远程入侵不在网络上传输数据包的主机相当困难，但是我们知道，这并非不可能。不是有人说过“不沉的泰坦尼克吗？”

第 18 章 Chapter 18

供程序员使用的日志

本章要点

- 角色和职责
- 程序员使用的日志记录
- 安全考虑因素
- 性能考虑因素

18.1 概述

你是一位程序员吗？是否有支持人员到你这里，询问你的应用程序为什么不能正常工作？程序员常常没有提供日志消息的足够信息，或者提供过多的信息。关键在于，要理解如何在日志中提供足够信息和安全及性能考虑因素之间求得平衡。此外，你的软件生成的日志消息可能由日志收集和分析工具收集。这意味着，必须理解和考虑日志的可解析性。本章介绍的概念可以应用到任何语言和环境，可以视为最佳实践。

18.2 角色与职责

在深入程序员使用的日志记录技术的细节之前，重要的是理解常见的角色和职责，因为它们与软件开发及编程相关。本节介绍的角色是较为常见的；你的组织中实际的角色可能有所不同。表 18.1 概述了常见的角色和职责。

表 18.1　内部角色和职责摘要（Chuvakin，2011）

角　色	职　责
业务 / 产品负责人	负责产品、特性集、应用程序等的个人或者集体
安全团队	为内部开发的软件开发日志记录标准；这一过程涉及业务负责人
IT/ 开发经理	强制实施日志记录标准，作为“必备”功能
IT 支持团队	支持生产应用程序：在应用程序表现异常时阅读日志文件
软件架构师	“获取”日志记录的价值，理解审计和调试日志之间的不同
软件开发人员 / 程序员	使用程序库、API 和日志记录功能，遵循日志记录标准

职责分离是保护生产环境所必需的安全原则。但是应该注意，在某些情况下，软件开发人员同时也是架构师和 IT 支持人员。这种情况一般发生在较小的组织内，它并不理想，但是确实会发生。说它不理想是因为当一个人身兼多职时，界线就变得模糊，判断和好的过程就无法实施。

职责分离的另一个重要性围绕监管依从性。各种监管实体都规定软件开发人员不能对生产系统有写入权限。在某些情况下，开发人员不能访问生产系统。那么，开发人员如何帮助生产环境的故障排除？下面是几条途径：

1）开发人员和生产支持人员一起，为操作人员提供指导。

2）可以使用屏幕控制（例如 Cisco 的 Web-ex，Microsoft 的 Office Communicator 等），开发人员从支持人员的终端上获得生产系统的一个临时视图和控制。

3）可以授予安全性例外。这在有合适的文档时一般是允许的。要点是开发人员可以在短期内（例如 30 分钟或者 1 小时）对系统有临时权限。

确保你理解开发人员在监管依从性环境中可以做什么，不能做什么。更多信息参见第 19 章。

你可能已经注意到，这些角色是面向内部的，不反映组织之外的角色。表 18.2 总结了组织之外的常见角色和职责，这通常是在商业和开源组织中的情况。

表 18.2　外部角色和职责摘要（Chuvakin，2011）

角　色	职　责
商业软件供应商	开发并采用日志标准
开源社区	创建日志记录 API、程序库等，然后用它们赢得声望和采纳
日志分析供应商	鼓励日志记录标准化，并利用标准消费数据，在产品中展示

18.3　程序员所用的日志记录

本节假定你是日志记录编程的新手。如果你已经精通了日志记录的相关概念，可以跳过这一部分。

现在，我们简单地看看程序员应该留意日志记录的常见原因。

- 程序员是日志消息的最终来源。确保程序生成有价值、实用的日志完全依靠程序员。格式良好和及时的日志消息对调试和诊断收集很有帮助，而且可以帮助程序员了解程序到底做了什么。
- 日志消息中提供合适的信息，有助于协助检测恶意行为。

认真考虑日志记录有几个具备说服力的理由。我们现在开始说明，为什么日志记录对于程序员很重要。但是作为程序员，日志记录背后是什么机制呢？我们先来看看这一过程的概要步骤。

1）程序员编写应用程序，决定在代码中记录哪些日志消息。

2）日志记录的列表至少应该由 IT 支持人员审核，以便得到他们的意见。将这些人员包含在内的第二个理由是你能够从监控的角度理解他们的需求。他们可能有你所不知道的特殊需求。

3）程序员编写代码，真正地完成有助于日志记录活动的函数或者方法调用。正所谓细节是魔鬼：

a）一般来说，有一个配置文件指明日志写入的位置。这包括本地磁盘、远程磁盘（网络文件系统（NFS）等）、syslog 等。

b）还要考虑日志轮转方案（详见后面的小节）。

4）日志被用于调试、分析等。

上面描述的是概略的过程。但是，应该记录哪些信息呢？我们将在下一小节探索这个主题。

18.3.1 日志应该记录哪些信息

在讨论这个主题前，理解什么因素使日志有用很重要。日志消息应该说明发生的情况和原因。回忆一下第 2 章中日志记录 5 个“W”的概念。我们快速地复习一下：

- 涉及谁？（Who）
- 发生了什么？（What）
- 发生在哪里？（Where）
- 发生在何时？（When）
- 为什么发生？（Why）
- 如何发生的？（How）

除了上述 6 个项目，以及第 6 个项目是以“H”(How，如何) 开头的以外，你还注意到什么？将这些项目应用到日志记录，很容易发现，五个“W”包含了有助于理解发生的情况和原因的所有必要组成部分。归纳一下，注意这五个“W”，就会发现实用的日志应该包含状态和上下文，状态可以是包含在变量、返回值、堆栈信息等的任何程序信息。上下文是有助于描述写入日志消息原因的支持信息。

表 18.3 提供了应该记录的日志消息类型的基本原则。

表 18.3　日志类型基本原则

应该记录的类型	描　述
AAA（身份认证、授权、访问）	成功和失败的身份认证或者授权决策；系统访问、数据访问和应用程序组件访问；以及远程访问，包括分布式环境中一个应用程序组件对另一个组件的访问
更改事件	系统或者应用程序更改（特别是权限更改）；数据更改（包括创建和销毁）；应用程序和组件安装与更改
“有害事件“/威胁	无效输入和其他可能的应用程序违规使用；其他已知会影响应用程序的安全问题
资源问题	资源耗尽、超过容量等；连接性问题；到达限值
混合可用性问题	系统、应用程序、应用程序模块或者组件的启动与关闭；故障和错误，特别是影响应用程序可用性的错误；以及影响可用性的备份成功和失败

表 18.3 是一个经验法则参考列表。这是很好的一组应该始终记录的日志消息类型。你的具体需求可能与列出的有所不同，但是知道基本的日志类型，就成功了一半。

警告　记录状态信息

在日志文件中记录某些信息会带来某些安全后果。这一主题的更多信息参见后面的 18.4 小节。

18.3.2　程序员使用的日志记录 API

提供任何日志记录 API 的细节超出了本章的范围。作为替代，表 18.4 概述了常用的 API。

表 18.4　常见语言的日志记录库摘要

语　言	库
Java	Log4j (http://logging.apache.org/log4j/)
	Logback (http://logback.qos.ch/)
Perl	Log4Perl (http://mschilli.github.com/log4perl/)
C/C++	Log4C (http://log4c.sourceforge.net/)
Unix / Shell	Logger (man logger)

我们来举一个简单的例子。Logback 是流行的 Java 日志记录 API。你可以用配置文件控制其日志记录行为，真正的思路就在于此。下面是一个 Logback 配置文件的例子：

```
<configuration>
   <appender name="STDOUT" class="ch.qos.logback.core.
    ConsoleAppender">
      <!-- encoders are assigned the type ch.qos.logback.classic.
            encoder.PatternLayoutEncoder by default -->
   <encoder>
      <pattern>
      %d{HH:mm:ss.SSS} [%thread]%-5level%logger{36} -%msg%n
</pattern>
      </encoder>
```

```
    </appender>
    <root level="debug">
        <appender-ref ref="STDOUT" />
    </root>
</configuration>
```

上面的例子将日志记录到标准输出（STDOUT），是最简单的例子。

利用上述配置，从应用程序中用库生成的日志消息格式如下：

```
17:44:58.828 [main] INFO chapters.configuration.MyApp2 - Entering
    application.
```

配置中驱动这一格式的是下面这一行：

```
%d{HH:mm:ss.SSS} [%thread]%-5level%logger{36} -%msg%n
```

我们按照顺序看看每个参数：

- %d{HH:mm:ss.SSS}：%d 输出日志事件的日期。左右花括号告诉 Logback 接受 ISO 8601 日期格式。
- [%thread]：左右括号是字面量字符串，意味着它们将会出现在日志消息中。%thread 将打印线程名（如果设置的话）或者调用日志记录消息的方法。
- %-5level：输出日志记录级别并左对齐，最小宽度为 5，没有最大宽度。
- %logger{36}：这个参数打印日志记录器，最多 36 个字符。本例中日志记录器就是应用程序本身。
- %msg%n:%msg 是提供给日志记录（由程序员提供）调用的原始日志消息。%n 是平台相关的行分隔符。

这一配置很容易理解，这是使用 Logback 之类的 API 的关键优势。Logback 的另一个很好的特性是可以配置为在配置文件变化时发出通知。这很有用，因为你没有必要停止和重启应用程序。首先，API 检测到更改，然后将更改加载到文件，在运行中的应用程序里自动开始使用修改后的配置。这一特性通过在 configuration 块中使用 scan 参数来实现：

```
<configuration scan="true">
...
</configuration>
```

默认扫描周期是 1 分钟。这在你希望更改日志消息格式时很方便。你所必要做的就是编辑该文件，做出更改并保存。Logback 基础设施负责其余的工作。你可以“即时”做出更改，而没有必要重新编译代码或者重启应用程序。

Logback 的完整细节请参见 Logback 手册。

18.3.3 日志轮转

日志轮转值得稍作讨论。虽然程序员通常不需要负责日志轮转，但是如果你对此有所理

解，将是很好的事情。首先，日志轮转机制有两种基本类型：

1）使用日志轮转脚本管理日志何时轮转，何时存档到某个外部存储位置。

2）应用程序本身负责处理日志轮转，这可以通过定制编写的应用程序代码，或者使用第三方日志记录库的内建功能处理。

但是，什么是日志轮转？日志文件轮转方案中，活动日志文件被移到存档副本，并为应用程序创建一个新的空文件，开始写入。这种技术在日志文件保存多个拷贝，用于短期分析和场外存档及存储时很有价值。

从应用程序的立场出发，它并不知道日志轮转的发生。可以采用多种日志轮转方案：

- 基于时间：日志文件根据某个时间周期轮转，例如每个小时、每天、每周等。
- 基于大小：日志文件在到达某个预定尺寸时轮转，例如 10MB、100MB 等。
- 基于大小和时间：这种方案结合了基于时间和基于大小的方案。日志文件根据时间存档，但是每个日志文件限制在某个预定尺寸之内。

我们来简单地看看 Logback 的基于时间和基于大小轮转配置：

```
<configuration>
   <appender name="ROLLING" class="ch.qos.logback.core.rolling.
    RollingFileAppender">
      <file>mylog.txt</file>
      <rollingPolicy class="ch.qos.logback.core.rolling.
        TimeBasedRollingPolicy">
         <!-- rollover daily -->
         <fileNamePattern>mylog-%d{yyyy-MM-dd}.%i.txt</fileNamePattern>
         <timeBasedFileNamingAndTriggeringPolicy
               class="ch.qos.logback.core.rolling.
                SizeAndTimeBasedFNATP">
            <!-- or whenever the file size reaches 100MB -->
            <maxFileSize>100MB</maxFileSize>
         </timeBasedFileNamingAndTriggeringPolicy>
      </rollingPolicy>
      <encoder>
         <pattern>%msg%n</pattern>
      </encoder>
   </appender>

   <root level="DEBUG">
      <appender-ref ref="ROLLING" />
   </root>
</configuration>
```

这个配置文件示例将在每天或者日志达到 100MB 时轮转。这对于记录大量消息的应用

程序来说是实用的技术。不等到日志文件增长到 1GB 就进行轮转，就能得到可管理（也可搜索！）的日志数据。

18.3.4　不好的日志消息

如果不谈及日志消息的坏习惯，对于有关程序员日志记录的章节来说就不完整。首先，我们来看一些不好的日志消息（Chuvakin，2010）。第一条消息包含了遗漏了年的时间戳：

```
Aug 11 09:11:19 xx null pif ?exit! 0
```

上述消息还遗漏了时区信息和其他有意义的内容。

下面看到的消息不仅完全遗漏了时间戳，还包含正在过渡到某个权限级别的人的用户名：

```
Message 202 User transitioning priv level
```

“魔法”或者秘密数字（本例中的 202）的使用很不好。如果没有关于该数字含义的文档，它就不应该包含在日志条目中。

最后，日志记录中常常会看到下面这样的例子：

```
userenv[error] 1040 XYZCORP\wsupx No description available
```

这位程序员直率得可怕：她决定放弃，告诉我们这里没有任何说明。

表 18.5 概述了你应该如避瘟神的常见坏习惯。

表 18.5　日志记录坏习惯摘要

坏习惯	解　释
遗漏时间戳和时区	没有这些信息，难以知道日志实际发生的时间，这可能损害调查规程、数据搜索等
神秘数字	日志消息中常常出现神秘的数字。问题是许多时候这些数字没有支持的文档。这不仅导致误导日志审核人员，而且令他们感到沮丧
描述模糊或者没有描述	日志条目必须清晰、简洁和容易理解。模糊或者遗漏描述不仅使人们和自动化工具难以解读消息，在调查系统停机或者潜在安全问题时还会浪费宝贵的时间
没有源 / 目标 IP/ 主机名和端口	不是所有应用程序都面向连接。但是对于客户 - 服务器系统，源和目标 IP 以及端口信息必须包含在日志条目中
没有唯一的消息标识符	每条日志消息有唯一的标识符很重要。这个 ID 通常是递增的整数值。拥有唯一描述某条消息的方法，对于日志消息的搜索和其他应用程序很有价值。许多编程语言有预定义的例程以生成全局唯一标识符（UUID）。尽管如此，唯一消息标识符通常由解析日志消息的系统或者工具生成
没有唯一的消息类型标识符	这不同于唯一的 ID，通常是标识消息所述类型或者级别的字母数字值。解决这个问题的方法之一是将日志消息的常见部分连接起来

18.3.5　日志消息格式

最佳的日志消息适合于人工和自动分析。日志消息的各个组成部分放在一起会有更重要的含义。下面的列表提供了实用的日志条目中应该包含的部分：

- 用户名有助于回答与用户或者管理员活动相关的活动是由“谁”发起的。此外，包含身份提供者的名称或者担保用户名的安全领域也很有帮助（如果这些信息可用的话）。
- 对象有助于指出受到影响的系统组件或者对象（如用户账户、数据资源或者文件），回答“什么”的问题。
- 状态也有助于回答“什么”的问题，它说明针对对象的操作成功还是失败。（也可能有其他类型的状态，例如“延期”）
- 系统、应用程序或者组件有助于回答“何地”的问题，必须提供相关的应用程序上下文，例如发起者和目标系统、应用程序或者组件。
- 对于和网络连接性或者分布式应用程序操作相关的消息，源有助于回答“来自何处”的问题。源可以是互联网协议（IP）地址或者主机名的形式。应该使用的相关组件取决于你的应用程序，包括目标、源端口和目标端口。
- 时间戳和时区回答“何时”的问题。时区对于分布式应用是不可或缺的。除了时间戳和时区，某些高容量系统还使用事务 ID。
- 原因有助于回答“为什么”的问题，这样日志分析就不需要深入地挖掘潜在的原因。记住，日志的客户是安全和审计人员。
- 操作有助于回答“如何”的问题，它提供了事件的特性。
- 优先级有助于表明事件的重要性。但是，按照重要性对事件进行统一的评级是不可能的，因为不同的组织有不同的优先级。（例如，不同公司对于信息可用性和机密性有不同的策略）
- 唯一的会话 ID 有助于集合不同线程和进程中的相关消息。
- 进程 ID（PID）和线程 ID（TID）对日志都很有用，可以关联运行应用程序及其日志记录。如果你的应用程序写入与其他应用程序共享的日志文件，这就很实用。
- 活动度量不是所有应用程序都需要记录的组件。例如，如果你的应用程序代表某个人或者其他应用程序，将数据从一个位置传输到另一个位置，对这个组件的日志记录可以由分析系统使用，检测何时试图传输超过预期大小的数据批次。

好了，那么根据上述原则，实用的日志消息是什么样子呢？我们来考虑下面这条虚构的消息：

```
2010/12/31 10:00:01AM GMT+7 priority=3, system=mainserver,
module=authentication, source=127.0.0.1, user=kjschmidt(idp:solar),
action=login, object=database, status=failed, reason="password
   incorrect"
```

我们看到，这条消息有刚刚讨论的许多组件。每个组件使用了名称 / 取值配对。这种常用的技术使用等号创建一个名称和取值的映射。然后，用逗号作为分隔符，标记名称 / 取值配对之间的分隔。你还应该看到，对于如何解析消息没有任何歧义。你首先从逗号处分离，得到一个名称 / 取值配对的列表。然后，从等号处分离，得到每个名称 / 取值配对为你提供的名称及取值。

为了得到圆满的解释，我们应该用五个“W”来检查日志消息，看看它是否能够经受严格的考验。现在我们就这样做：

- 涉及谁：kjschmidt
- 发生了什么：failed authentication（身份认证失败）
- 发生在哪里：database（数据库）
- 发生在何时：2010/12/31 10:00:01AM GMT+7

分隔符

虽然逗号常被用作分隔符，但是需要小心。如果预测到名称 / 值配对中包含逗号，选择不同的分隔符就更加明智了。例如，如果你有 reason=“1, 2, 3,”这样的数据，数字之间的逗号可能导致解析机制出现问题。

消息类型唯一标识符

回忆表 18.5 中的一个坏习惯“没有唯一消息标识符”。如果我们使用本节介绍的虚构日志消息，采用“对象 – 操作 – 状态”方式，可以给出“database-login-failed”这样的标识符，目标是创建一个可以用于唯一标识日志消息中发生的情况的消息标识符。记住，其他日志消息可能有同样的唯一消息标识符，但是参数不同（比如用户名、系统等）。

- 为什么发生：Password incorrect（密码错误）
- 如何发生的：The user entered a bad password（用户输入错误的密码）

可以看到，这种格式可以轻松地回答五个“W”的问题。这个例子说明了最终的目标：清晰地理解发生的情况，以便及时地解决。

18.4　安全考虑因素

有些信息绝不应该记录到日志消息中。任何可以用于标识人、财务记录、生日和其他类似情况的数据都不应该记录。基本上，任何个人标识信息（PII）都应该禁止记录，这包括：

- 密码（数据库、用户等）
- 社会保险号码
- 生日
- 电话号码
- 全名
- 信用卡号码
- 驾驶证号码

- 基因信息
- 保险信息
- 任何类型的标识号码
- 生物计量学信息

你应该了解了吧。有些程序员自作聪明地将日志消息格式化为下面的形式：

```
…Database login failure for user=Bob, password=XXXXXX
```

为什么要记录都是”X”的密码参数呢？这是浪费时间和空间。一定要抵制这样编写日志消息的诱惑。

18.5 性能考虑因素

现在，你应该询问自己，日志记录对应用程序的性能有什么影响。考虑下面这段从数据库中读取记录并进行处理的伪代码：

```
While(resultSet) {
   Log("Getting record..\n");
   Record = getRecord();
   Log("About to process record\n");
   processRecord(Record);
   Log("Done processing record\n");
}
```

如果我们每次处理的数据库结果集中有几百个记录，那么影响不大。但是，如果每个结果集有几千个、几万个甚至几十万个会怎么样呢？如果我们每分钟都有这么多记录数据呢？每当我们调用 Log() 方法，它实际上都将打开日志文件并写入日志消息。现在，我们没有考虑操作系统是否缓存输出。假设我们每次调用 Log() 时，数据都直接写入日志文件。这意味着对于每个记录，我们要写入 3 条日志消息。将其扩大到数十万条记录，数量就很大了。与记录程序执行的每个操作相比，建立一个日志记录级别方案是更好的想法。

大部分日志记录 API 允许你指定实际日志消息的记录级别。日志级别通常包括 INFO、DEBUG、WARN、ERROR 和 FATAL，每个级别的含义都相当明显。这些 API 一般还允许你开启和关闭日志级别；这在配置级别上完成。表 18.6 是日志级别方案的一个例子。

表 18.6　日志级别方案示例

日志级别	描述	默认记录?
INFO	INFO 用于记录常规处理消息和应用程序状态	否
DEBUG	这个级别用于标识帮助开发人员调试的日志消息	否
WARN	WARN 级别用于引起对可能有害的日志消息的注意	是
ERROR	ERROR 是应用程序操作异常的情况，但是不一定导致应用程序失败	是
FATAL	FATAL 级别是严重的情况，会导致应用程序终止	是

日志记录级别的思路是了解默认情况下希望记录的日志级别阈值，以及出现问题时应该“开启”哪个级别。例如，如果一个应用程序运行不正常，应该按照如下步骤处理：

1）在配置文件中启用 DEBUG 日志记录级别。

2）允许应用程序在一段时间内写入调试级别日志消息。

3）在配置文件中禁用 DEBUG 日志记录级别。

这样你就可以在需要的时候收集日志消息，同时平衡确保应用程序最优化运行的需求。

还要注意，日志消息越大，占据的磁盘空间也就越大。但是，现在的磁盘空间相当便宜，可能有海量的磁盘空间供你定期存档应用程序日志文件。

18.6 小结

我们可以用如下的经验法则来总结本章：

1）确保应用程序生成清晰、简洁、可解析的日志消息。

2）理解日志轮转。

3）理解日志消息中不能放入的信息。

4）理解基本性能考虑因素和处理的方式。

参考文献

Chuvakin, A. (2011). Application logging good bad ugly ... beautiful? *Upload & Share PowerPoint Presentations and Documents*.Web, August 5. <http://www.slideshare.net/anton_chuvakin/application-logging-good-bad-ugly-beautiful-presentation>.

Chuvakin, A. (2010). How to do application logging right. *Building Security In*. Web. <http://arctec-group.net/pdf/howtoapplogging.pdf>.

Logback Manual. Web, August 5, 2011. <http://logback.qos.ch/manual/>.

Chapter 19 第 19 章

日志和依从性

本章要点

- PCI DSS
- ISO 2700x 系列
- HIPAA
- FISMA

19.1 概述

日志虽然常常得不到 IT 经理的正确评价，但是能够为安全管理提供有用的信息。但是，获取日志数据需要花费时间和精力，这两者在 IT 组织中往往供应不足。首先，这些工作看起来很乏味，因为数据的容量很大且具有主观性。日志越来越不仅是系统管理员的数据来源，这也增加了其战略价值。日志记录和通过日志管理软件或者其他工具跟踪此类活动，是实现 IT 可审核性的主要手段，因为大部分用户和系统操作可能记录在日志中。实现组织中的可审核性可能需要许多其他手段，但是日志是遍及各个 IT 领域的一个机制，甚至延伸到技术的范围之外。如果你的 IT 运营不可审核，也就意味着你的业务无法审核。

如果你的组织没有认真地对待日志，那么就可以说明你对 IT 的可审核性并不重视。这也就是日志记录成为一种完善的依从性技术，许多法规和法律（如 PCI DSS、FISMA、HIPAA）以及最佳实践框架（如 ISO2700 和 COBIT）都对此作出强制性要求的原因。

如前所述，所有 IT 用户，不管是恶意的还是好的企业公民，都会在各种日志中留下活

动的踪迹。这些数字指纹由许多 IT 组件（如用户拥有的桌面电脑、服务器和防火墙、路由器、数据库和业务应用程序）生成。这些记录随着时间的推移而积累，造成了堆积如山的不同类型的日志数据。

总体来说，法规在日志数据上有如下的某些或者全部要求：

- 上述法规的典型日志记录、日志管理和安全监管需求包括：
- 有充足的日志记录。各种法规在“充足”的含义上有显著的不同。有些法规只规定组织有审计日志记录。
- 集中收集日志。有些法规要求收集日志并集中存储和分析。
- 审核日志数据。许多法规中最困难的部分是强制日志审核。例如，PCI DSS 要求每天审核来自范围内系统的日志。很明显，这并不意味着每个单独的日志条目都必须人工阅读。
- 留存日志一段时间。法规要求各种不同的日志留存期——从几个月到几年。有些法规只要求组织必须有日志留存策略，而没有规定具体的数字。

监控安全性。有些法规规定了网络和 Web 警报的审核，在必要时可以部署一个事故响应过程。其他任务包括日志数据保护、时间同步等。（来源：http://searchcompliance.techtarget.com/tip/Log-management-and-compliance-Whats-the-real-story）

“法规只强制要求拥有日志数据”是常见的错误概念。

下面我们来研究几种流行的法规，看看它们与日志记录、日志分析和日志管理的相关性。

19.2 PCI DSS

本节介绍 PCI DSS 日志记录基本知识及其要求。

总体上，日志记录和监控不只限于需求 10，而是遍及 PCI DSS 的全部 12 个需求；在 PCI DSS 中强制日志记录和监控的关键领域是需求 10，和需求 11 及需求 12 的一些部分。

关键需求 10

确切地说，需求 10.1 覆盖了“建立一个过程，将对系统组件的所有访问（特别是用根用户等管理性权限进行的访问）与每个单独用户联系起来。”这是一个有趣的需求；它强制的不是日志的存在或者建立日志记录过程，而是提到日志必须与单独的个人绑定（不是计算机或者生成日志的“设备”）。这个需求常常给 PCI 实施者造成问题，因为许多人将日志视为“人的操作记录”，而实际上他们所拥有的只是“计算机操作的记录”。将后者映射到实际的用户，往往带来额外的挑战。顺便说一句，PCI DSS 需求 8.1 强制组织“在允许访问系统组件或者持卡人数据之前，为所有用户分配一个唯一的 ID。”，这有助于使日志更加实用（来源：http://chuvakin.blogspot.com/2010/11/complete-pci-dss-log-reviewprocedures_30.html）。

接下来，10.2 小节定义了需要记录（或者‘使事件可以重建”）的系统事件最小列表。这

种需求的动机是评估和监控用户操作，以及影响信用卡数据的其他事件（如系统故障）。（来源：http://chuvakin.blogspot.com/2010/11/complete-pci-dss-log-review-procedures_30.html）。

下面是来自 PCI DSS（2.0）的需求列表（必须记录的事件）：

10.2.1　对持卡人数据的所有单独用户访问。

10.2.2　具有根或者管理权限的任何个人进行的所有操作。

10.2.3　对所有审计跟踪的访问。

10.2.4　无效的逻辑访问尝试。

10.2.5　标识和身份认证机制的使用。

10.2.6　审计日志的初始化。

10.2.7　系统级对象的创建和删除。

可以看到，上述需求覆盖了数据访问、特权用户操作、日志访问和初始化、失败和无效的访问企图、身份认证和授权决策以及系统对象更改。重要的是，这一列表根植于 IT 治理“最佳实践”，规定了访问、身份认证、授权变更管理、系统可用性以及可疑活动的监控。

而且，PCI DSS 需求 10 更加深入，涵盖了每个事件必须记录的具体数据字段或者取值。

这些字段提供了健康的最低要求，各种 IT 平台中的日志记录机制一般都能超过该要求。

规定的字段如下：

10.3.2　用户标识。

10.3.2　事件类型。

10.3.3　日期和时间。

10.3.4　成功或者失败的指示。

10.3.5　事件来源。

10.3.6　受影响的数据、系统组件或者资源的标识或者名称（来源：www.pcisecurity standards.org 中的 PCI DSS）

可以看到，这个最小列表包含了事故分析需要的基本属性，可以回答如下问题：何时、谁、何地、发生何事、来自哪里。例如，如果试图发现谁修改了信用卡数据库，将所有事务细节复制到一个隐藏文件（典型的内部人员权限违规使用），了解上述所有记录就非常有用。

下一个需求 10.4 描述了常被忽视但十分关键的需求：所有日志中具有准确和一致时间的需求。时间和安全事件监控的联系似乎相当简单。在家庭或者小型办公网络中，系统时间常常被发现是随意设置的。不管你的服务器设置成什么时间，如果你设计的网络需要一定的可靠性，系统就应该配置为从可靠来源获得时间同步，例如与网络时间协议（NTP）服务器同步。

接下来，人们必须处理日志的机密性、完整性和可用性（CIA）。PCI DSS 的 10.5.1 小节介绍了机密性：“将对审计跟踪的查看限制在与工作相关的需求上”。这意味着，只有必须查看日志才能完成工作的人才能够查看日志。原因之一是和身份认证相关的日志总是包含用户名。虽然不是真正的秘密，但是用户名称信息提供了密码猜测所需的 50% 信息（另外 50% 是密码）。而且，因为用户错误输入证书，而在日志中显示密码的情况并不少见。编写质量

低劣的 Web 应用程序可能造成密码和 Web 统一资源定位符（URL）一起记录在 Web 服务器日志中。

接下来是“完整性”。根据 PCI DSS 的 10.5.2 小节，必须“保护审计跟踪文件免遭未授权修改”。这是显而易见的，因为如果日志被未授权方（或者任何人）修改，它们就不再是系统和用户活动的客观评估跟踪。

但是，人们不仅必须保留来自恶意用户的日志，还应该保留系统故障和系统配置错误结果的日志。这涉及日志数据的“可用性”和“完整性”。PCI DSS 10.5.3 明确指出，必须“立即将审计跟踪文件备份到集中日志服务器或者难以修改的媒介”。确实，将日志集中到一个或者一组用于日志分析的服务器，对于日志保护和增加日志实用性都是不可或缺的。将日志备份到 CD 或者 DVD（或者磁带）是这一需求的另一种结果。人们应该始终记住，磁带上的日志不容易访问，在出现事故时也无法搜索。

许多网络基础设施（如路由器和交换机）被设计为将日志放到外部服务器，在设备自身上只保留极少量（或者没有）日志。因此，对于这些系统来说，集中化的日志最为关键。PCI DSS 的需求 10.5.4 说明了这种需求：“将无线网络的日志复制到内部 LAN 上的日志服务器中。”

为了进一步降低日志修改的风险，并证明这样的修改没有进行，需求 10.5.5 要求“在日志上使用文件完整性监控和变更检测软件，确保现存日志数据不能在没有警告的情况下更改”。与此同时，在日志文件中添加新的日志数据不应该生成警报，因为日志文件不断增长，不会自己压缩（除非日志轮转或者存档到外部存储）。文件完整性监控系统使用加密哈希算法，将文件和一个已知的正确副本比较。问题是，日志文件因为新记录的添加而不断增长，从而导致完整性检查的削弱。为了解决这一矛盾，应该注意完整性监控只能保证没有被日志记录组件不断写入的日志的完整性。

下一个需求最为重要，也是最容易被忽视的。许多 PCI 实施者忘记了，PCI 需求 10 不仅要求“有日志”，还要求“有日志，并且查看它们。”10.6 小节明确说明 PCI 组织应该按照 PCI DSS，“至少每天一次审核所有系统组件的日志。日志审核必须至少包含执行 IDS 和 AAA 服务器（例如 RADIUS）等安全功能的服务器”。本章余下的部分介绍详细的日志审核规程和常规做法。

因此，PCI DSS 需求还覆盖了需要“每天审核”的日志源范围，而不仅是需要配置日志、保留或者集中化日志。考虑到大型 IT 环境每天可能生成以 GB 计算的日志，人工阅读所有日志是不可能的。因此，PCI DSS 在这个需求上有个注解：“可以使用日志收集、解析和警报工具实现和需求 10.6 的依从性”。

最后一个需求（10.7）处理另一个非常重要的日志记录问题——日志留存，它规定：“留存审计跟踪历史至少一年，保持在线可用性至少三个月”。和其他无数需求不同，这一需求直接处理复杂的日志留存问题。因此，如果你无法查看 1 年前的日志，就违反了规定。而且，PCI DSS 在更新后的版本 1.1 中还明确地增加了一年的需求规定。

下面，我们来总结一下目前为止学习的 PCI 日志记录需求：

- PCI 需求 10 规定以预先定义的细节水平记录来自范围内所有系统的特定事件。
- PCI 要求日志中所有操作与实际用户挂钩。
- 范围内的系统上的时钟和时间应该同步。
- 应该保护所有收集日志的 CIA（机密性、完整性、可用性）。
- 日志应该定期审核；特定的日志应该至少一天审核一次。
- 范围内的所有日志应该留存至少一年。

现在，我们做好准备，将要更加深入地研究日志和监控，这些日志和监控不仅在需求 10 中规定，还出现在所有其他 PCI 需求中。许多人可能认为 PCI 中的日志都在需求 10 中说明，但实际情况更为复杂：日志出现和隐藏在所有其他的小节中（来源：http://chuvakin.blogspot.com/2010/12/complete-pci-dss-log-reviewprocedures.html）。

几乎所有满足 PCI DSS 需求的规定（如数据加密或者防病毒更新），都可能使用日志文件来证明。

例如，需求 1“安装和维护防火墙配置，以保护持卡人数据”中提到，组织必须有“批准和测试所有外部网络连接以及防火墙配置变更的正式过程”。但是，在建立这种过程之后，人们必须验证防火墙配置变更确实得到授权，并且按照书面的变更管理过程进行。这时日志记录变得极其方便，因为它说明了实际发生的情况，而不是“应该发生”的情况。

需求 1.3 包含防火墙配置的指导原则，有关于入站和出站连接性的具体陈述。必须使用防火墙日志来验证这些配置，仅仅审核配置还不够，因为只有日志才能说明“实际发生了什么”而不只是“如何配置”。

类似地，需求 2 讨论密码管理“最佳实践”和一般的安全加固，例如不运行没有必要的服务。日志可以说明何时启动了之前禁用的服务，这可能是不知情的系统管理员或者攻击者的操作造成的。

需求 3 更进一步，处理数据加密，这和日志记录有着直接而明确的联系。例如，3.6 小节隐含了一个要求：拥有验证这些活动是否真正进行的日志。确切地说，大部分加密系统记录密钥生成、分发和废止的日志，这些日志对于满足上述需求是至关重要的。

需求 4 处理的也是加密，由于同样的原因，和日志记录也有隐含的联系。

需求 5 针对病毒防御。当然，为了满足 5.2 小节的需求——“确保所有防病毒机制最新、活跃地运行，并且能够生成审计日志”，必须查看所提到的这些日志。

所以，即使是“使用和定期更新防病毒软件”也可能在评估期间需要请求日志数据，因为这些信息由防病毒评估日志提供。著名的“防病毒更新失败”问题也在日志中反映，它说明公司正处于恶意软件的威胁之下，因为没有更新最新特征码的防病毒软件只能造成虚假的安全感，损害依从性工作。

需求 6 也处于同一范畴：它要求组织“开发和维护安全的系统和应用程序”，没有强大的评估日志记录功能和应用程序安全监控，这是无法想象的。

需求 7 规定“根据业务按需了解的原则限制对持卡人数据的访问”，需要日志来验证谁能够访问文中所指的数据。如果不应该看到数据的用户出现在访问数据的日志文件中，就需要采取纠正措施。

为每个访问系统的用户分配唯一的 ID 符合其他安全“最佳实践”。在 PCI 中，这不仅是一个“最佳实践”，还是一个需求（需求 8“为每个具有计算机访问权的个人分配一个唯一的 ID”）。显然，人们必须“控制用户 ID、证书和其他标识符对象的添加、删除和修改”(PCI DSS 8.5.1 小节)。大部分系统记录这些活动的日志。

除此之外，8.5.9 小节“每 90 天至少更改用户密码一次”，也可以通过审核来自服务器的日志来验证，以便确保所有账户在 90 天内至少修改一次密码。

需求 9 介绍了安全的一个新领域——物理访问控制。甚至连包含访问者日志维护（可能以物理日志簿的形式）的 9.4 小节也和日志管理有关（如果这些访问者日志是电子形式的）。对于这类日志有单独的数据留存需求：“使用访问者日志维护访问者活动的物理评估跟踪。留存这些日志至少 3 个月，除非法律有其他限制”。

需求 11 说明扫描（或者“测试”）范围内系统漏洞的必要性。但是，在 11.4 小节中要求使用 IDS 或者 IPS：“使用网络入侵检测系统、基于主机的入侵检测系统和入侵预防系统，监控所有网络流量，并向有关人员发出可疑入侵的警报。保持所有入侵检测和预防引擎最新”。入侵检测只在审核日志和警报时才有用。

需求 12 涵盖了更高级的问题——安全策略、安全标准和日常运营规程（例如，需求 10 中强制的每日日志审核规程应该在这里反映）。但是，它也和日志记录相关，因为评估日志记录应该是每个安全策略的一部分。除此之外，事故响应需求也和日志记录相关：“建立、记录和发布安全事故响应和升级规程，确保及时和有效地处理所有情况”，如果没有高效的日志数据收集和及时审核，上述要求难以想象。

因此，PCI DSS 中的事件日志记录和安全监控远不止于需求 10。只有通过精心的数据收集和分析，才能满足 PCI 的广泛需求。

19.3 ISO 2700X 系列

ISO 27001 属于 ISO 27000 标准家族。它是由国际标准化组织（ISO）发布的一个信息安全管理系统（ISMS）标准，全名为 ISO/IEC 27001:2005“信息技术——安全技术——信息安全管理系统——需求。”该规范可以在 http://www.27000.org 和其他位置找到（很遗憾，要付费）。

我们将把重点放在与日志记录和监控相关的 ISO 控制上。这些内容集中在“A.10.10 监控”小节，其目标是：“检测未经授权的信息处理活动。”这与 ISO 27001 框架的安全任务保持一致，强调安全问题的检测，而非调查。

ISO A.10.10.1 小节“审计日志记录”有如下陈述：“应该生成记录用户活动、异常和信

息安全事件的审计日志，并保存一定的周期，以辅助未来的调查和访问控制监控。”这规定要拥有日志，而且在一个预先定义的时间周期（在你的日志留存策略中定义）内留存。它还简短地强调了日志中需要记录的事件类型，遵循本章引言中提到的传统依从性相关日志类别。

扩展上述定义，并加入 A.10.10.4 小节“管理员和操作员日志”（“系统管理员和系统操作员活动应该记入日志”）和 A.10.10.5“故障日志”（“故障应该记录、分析并采取相应的行动”）中的附加需求，我们可以总结 ISO 环境中所有系统需要记录的事件类型，如表 19.1 所示。

表 19.1　在 ISO 环境中需要记录的日志事件类型

日志消息分类	在 ISO 27000 日志记录和监控中的目的
身份认证、授权和访问事件，如成功与失败的身份认证决策（登录）、系统访问、数据访问和应用程序组件访问；特别是远程访问	跟踪和调查授权用户以及攻击者和恶意内部人员对系统的访问
更改（如系统或者应用程序变更，特别是权限变更）、数据变更（包括创建和销毁），以及应用程序和组件安装及更新	监控可能将系统暴露在攻击之下的更改，以及攻击者和恶意内部人员所做的更改
可用性问题，如系统、应用程序和应用程序模块或者组件的启动和关闭，以及故障和错误，特别是影响应用程序可用性和数据安全性的错误	监控运营问题和安全相关故障，以及系统业务可用性的损失
第 4 个类型是资源问题，涵盖了资源耗尽、超过容量和网络连接可用性问题以及达到各种限值等范围。所有这些日志能够为安全事故（如数据窃取）提供有用的初始证据	检测由于安全性和运营原因造成的资源问题
已知威胁，如无效输入和其他可能的应用程序违规使用、利用和已知攻击	检测并阻止攻击，以及对其后果的调查

所有这些事件可以提供对安全事故（如数据窃取）有用的初始证据。

在 Windows 上，这些事件包括了从登录到策略更改，再到应用程序更新及用户对数据的操纵等广泛的事件。在网络设备上，将包含安全性和可用性问题。

对于特权用户的监控（如“系统管理员和系统操作员活动”）应该加以特别注意。这种日志记录实际上是在 ISO 计划指导下创建的最重要的日志，因为它们可以作为全能 IT 管理员的关键可审核性手段。但是，为了用于这样的目的，日志必须通过日志管理解决方案，脱离这些管理员的控制。

上述几个小节还包含了日志留存和调查的其他需求。和 PCI DSS 不同，ISO 没有明确规定 1 年的日志留存期，只是提到日志留存，要求有单独的策略定义日志留存期。实际上，保留安全相关日志 1 年是常见的行业惯例。建议所有组织在整个环境中对物理和虚拟组件都实施这样的日志留存。

还要注意 10.10.5 小节中的“采取行动”需求。这提示我们不仅要收集日志，还要定义监控、调查和异常跟踪的策略及规程。

ISO 27002 提供实践中实施 ISO 系列标准的其他有用细节。例如，10.10.1“审计日志记录”

附加的原则规定，“特权的使用”和“系统工具及应用程序的使用”必须记入日志。

下一小节 A.10.10.2“监控系统使用”中明确强制规定“应该建立监控信息处理设施使用情况的规程，活动监控的结果定期进行审核。”这要求一个日志记录策略和监控的操作规程。ISO 还暗示着这些策略和规程的持续审核。例如，对于 ISO 项目范围内的系统和应用程序，操作规程必须包含定期或者实时日志审核工作流。这种审核由应用程序管理员或者安全管理员用自动化工具或者人工进行（但是由于收集和留存的日志体积，后者在较大的环境中不可能进行）。

ISO 27002 建议“单独设施的监控级别应该通过风险评估确定”，这为安全监控活动提供了合理的基础，不会缺乏或者过度控制。有趣的是，“I/O 设备连接 / 断开”明确地成为监控目标。

总体来说，这些关键控制要求创建一个日志记录策略，定义日志记录、日志收集和留存以及审核和分析——还可能有实时警报。

ISO 27001 包含了加固日志，使之免遭未授权修改和查看的主题。A.10.10.3 小节“日志信息保护”中规定“日志记录设施和日志信息应该得到保护，免遭篡改和未授权访问。”注意日志保护的两个重点：访问控制（保持日志机密性）和完整性保护（保持日志完整性）。大部分现代日志管理和 SIEM 系统提供严格的基于角色访问控制，按照“按需了解”的原则，将法定操作限制在授权人员上。在此基础上，系统还采用加密机制进行完整性检查，检测任何可能的日志更改。作为最后手段，日志可以写入“一次性写入”媒介（如 DVD）或者网络备份存档。虽然没有明确提及，但是这种控制暗示，恶意的内部人员不应该能够使用日志，进一步实现其邪恶的目的——日志安全措施应该对外部攻击者和恶意（以及不谨慎的）内部人员都有效。

除此之外，组织必须监控“日志文件媒体的存储容量超限，造成记录事件的失败或者覆盖过去的日志记录”(ISO 27002)。这种日志数据的意外丢失是不能容忍的。

最后，ISO A.10.10.6 小节“时钟同步”规定“组织或者安全域内所有相关信息处理系统始终应该与公认的准确时间源同步。”

和 PCI DSS 一样，日志定时对于安全监控、取证和故障排除都很重要。确保所有收集和分析的日志有正确的时间戳，且保持相关的时区信息，是行业的最佳实践。为日志使用 NTP 或者其他可靠定时机制，以便保持事件的正确时序以及绝对时间。

19.4　HIPAA

1996 年的《 The Health Insurance Portability and Accountability Act 》（ HIPAA）概述了健康信息的相关安全和隐私标准，包括电子和物理信息。该法案的主要使命是“改进健康保险覆盖团体和个人市场的便利性和持续性，和健康保险和保健服务中的浪费、欺诈和违规使用做斗争”（ 1996 年 HIPPA 法案）。该法案的拷贝可以在 HHS 网站上获得（ http://www.hhs.gov/ocr/privacy/hipaa/）。

特别是，该法的第 2 条（Title Ⅱ）“避免保健欺诈和违规；管理简化；医疗责任改革”包含安全规则（2.3 节，包括“受保护的电子健康信息”（Electronic Protected Health Information，EPHI））和隐私规则（2.1 节，包括所有受保护的健康信息（Protected Health Information，PHI））。

HIPAA 最新的增强版本称为《Health Information Technology for Economic and Clinical Health Act》(HITECH 法案)。该法案寻求“促进健康信息技术的采用和有意义的使用”和“通过加强 HIPAA 规则的民事和刑事执行，部分解决与健康信息传输相关的隐私和安全忧虑。”（2009 年 HITECH 法案）。

和 PCI DSS 不同，HIPAA 本身没有涉及安全控制级别和所采用的技术。这要求 HIPAA 所影响的组织，也称为“涵盖实体”，遵循法规的精神和非其文字。还要注意的一点是，接受支付卡的保险公司和许多医院要同时遵守 HIPAA 和 PCI DSS。可以理解，不同组织中，实用性的范围可能不同，因为支付处理系统不应该保存患者的健康信息，反之亦然。然而，考虑符合两种法规的技术和管理控制是谨慎的做法，在短期和长期来说都能够节约金钱。

下面的 HIPAA 需求广泛适用于日志记录、日志审核和安全监控：

- 164.308 小节 (a)(5)(ii)(C)“登录监控”要求监控接触患者信息的系统的登录和访问。这一需求适用于失败和成功的“登录企图”。
- 164.312 小节（b）“审计控制”广泛涵盖了审计日志记录和处理敏感健康信息的系统上的其他审计跟踪。这条需求隐含了这些审计日志的审核。
- 164.308 小节 (a)(1)(ii)(D)“信息系统活动审核”规定了各种 IT 活动记录的审核，如日志、系统利用率报告、事故报告和其他安全相关活动的指示。

HIPAA 中的其他需求也可能影响日志记录（来源：http://www.prismmicrosys.com/EventSourceNewsletters-June10.php）。

上面的条款说明，和 PCI DSS 相比，HIPAA 中的日志记录和监控要求本身不能真正地帮助公司回答部署和运营日志记录和日志管理时所需回答的关键问题——从技术和策略 / 规程的角度看都是如此。

特别是，下面的问题没有答案：

- “审计控制”应该记录哪些信息？哪些活动和事件？每个活动或者事件的哪些细节？
- 日志记录是否应该集中收集？
- 日志应该留存多久？
- 应该审核哪些特殊“活动”？审核的频率如何？
- 应该如何执行安全监控和“登录监控”？
- 如何保护审计记录？

由于这种情况，我们往往注意到，HIPPA 日志收集和审核成为了各种规模组织的错误多发点。日志需求对某些公司来说可能很难，例如部署了复杂系统的组织或者缺乏时间、资金和专业性的小型商店。模糊的原则不能帮助组织积极地进行日志记录和日志审核。在此基础上，日志记录和日志审核的复杂性在环境中有定制应用程序（而不仅仅是 Windows 服务器或

者 Cisco 防火墙）时显著上升。尽管趋势是远离传统和定制应用程序，许多医疗数据仍然保存在自制的应用程序中，在这些应用中，日志记录的配置就像是噩梦。

除了上述问题之外，还有一个问题不清晰：这些控制适用于处理敏感健康数据的实际应用，还是适用于底层平台？（来源：http://www.prismmicrosys.com/EventSourceNewsletters-June10.php）。

幸运的是，NIST 出版物 800-66《An Introductory Resource Guide for Implementing the Health Insurance Portability and Accountability Act（HIPAA）Security Rule》，见 http://csrc.nist.gov/publications/nistpubs/800-66-Rev1/SP-800-66-Revision1.pdf）中介绍了 HIPAA 安全规则实施的附加细节。

NIST SP 800-66 提供了加固受保护的电子健康信息的详细日志管理需求（根据 HIPAA 安全规则）的指南。

NIST 800-66 的 4.1 小节描述了信息系统活动定期审核的需求，例如审计日志、信息和系统访问报告以及安全事故跟踪报告。这一小节提出问题（“审核进行的频率多高？”和“审计信息保存在哪里？（例如单独的服务器）”）而不是提供答案。

4.15 小节试图提供“审计控制”的附加指导。虽然力求提供方法和实施者需要询问的问题，如“监控哪些活动？”（例如，包含 EPHI 的文件或者记录的创建、读取、更新和删除）和“审计日志应该包含哪些内容？”（例如，用户 ID，事件类型 / 日期 / 时间），但是这一文档并没有真正解决关键的实施问题，换言之，它没有告诉涵盖实体必须怎么做才能实现依从性。

而且，4.22 小节指出，操作和活动文档必须留存至少 6 年，并将安全活动记录（如日志）是否被视作“文档”的讨论留给实施者。

推荐的策略建议，公司从信息安全活动审核策略和过程入手。使用 NST 800-66 中提出的问题，人们可以规划这些策略的范围：需求适用性、记录的活动和记录细节、审核规程、异常监控过程等。

下面的内容引自 NIST 800-66：

- 谁负责总体过程和结果？
- 审核进行的频率多高？
- 审核结果分析的频率多高？
- 组织对员工违规有何约束政策？
- 审核信息保存在哪里（例如单独的服务器）？

接下来，组织必须真正地为日志记录和日志审核实施上述过程。这能够确保在覆盖的系统上创建日志记录，并且有足够的细节（日志记录）。顺便说一句，这些细节可以借鉴对应的 PCI DSS 原则。而且，这一过程将创建规程，“定期审核信息系统活动的记录，如审计日志、访问报告和安全事故跟踪报告”（日志审核）。虽然每天日志审核不是必需的，但是如果为了 PCI DSS 进行了审核，它们就可以进行扩展，以覆盖 HIPAA 系统。

对此，NIST 800-66 建议：

- “开发相应的标准操作规程。
- 确定审计跟踪数据的类型和驱动意常报告所需的监控规程。
- 异常报告或者日志如何审核?
- 监控报告在哪里归档和维护?”

这时，组织才做好准备进入下一步，启动日志记录并开始持续的日志审核。

虽然 HIPAA 没有提供日志记录和日志管理的详细指南，但是为公司提供了一个机会，遵循规章的精神而不仅仅是文字。可以理解，一些组织可能要等到经受处罚和强制活动之后才会采取行动。对日志记录的这种短视的方法只会走到“坏孩子“的那一边，使得网络犯罪分子能够窃取我们最敏感的数据。

而且，HIPAA 本身没有规定实施的安全控制和技术的级别。这要求受到 HIPAA 影响的组织（也称作涵盖实体）遵循法规的精神而非文字。还要注意的一点是，接受支付卡的保险公司和许多医院要同时遵守 HIPAA 和 PCI DSS。可以理解，不同组织中，适用性的范围可能不同，因为支付处理系统不应该保存患者的健康信息，反之亦然。然而，考虑符合两种法规的技术和管理控制是谨慎的做法，在短期和长期来说都能够节约金钱。

虽然平台级别的日志记录对于保护敏感的健康信息很有用，且大部分健康信息存储在数据库中，由保健专用应用程序处理，如专业供应商制作或者内部开发（通过外包开发人员）的应用程序。

和平台日志记录相比，HIPAA 164.312 小节（b）中提及的审计控制同样适用于应用程序日志记录，甚至比平台日志记录有过之而无不及。这意味着，定制应用程序需要设计为具备充足的日志记录。现有应用程序的日志记录需要评估，看看是否有充足的日志记录，应该注意的是，许多遗留应用程序往往没有记录足够的事件细节，甚至完全忽略事件日志的记录。因此，在启动这种项目之前，确定组织中的哪些应用程序包含受保护的健康信息（PHI）和现有的日志记录级别及方法很有意义。

我们来定义一些原则，指明需要记录哪些日志，才能从精神上和字面上满足 HIPAA 安全需求和 NIST 800-66 HIPAA 说明。

粗略地讲，最好的审计日志告诉你发生了什么，包括何时、何地和如何，以及涉及的人。这样的日志适合于人工、半自动和自动化分析。理想状况下，它们可以在不包含生成它们的应用程序的情况下进行分析，也肯定不需要该应用程序的开发人员。在保健应用程序中，可能无法联系到这些开发人员，安全团队必须自己推进工作。从日志管理的角度看，日志可以集中起来供分析和留存。最后，它们不应该降低系统的速度，而且，如果用作取证的证据时，能够证明是可靠的（来源：http://www.prismmicrosys.com/EventSourceNewsletters-July10.php）。

这里需要两个主要定义：

- 首先，有些类型的活动或者事件总是必须记录。例如，身份认证决策、健康信息访问和系统变更总是应该出现在日志中。

- 其次，对于每类记录事件，为了解读它们（由人工或者自动化系统进行），有些特定细节是必备的。例如，每个日志应该有可靠的时间戳，每个与用户活动相关的日志应该包含用户名。

还应该注意，某些细节绝不应该记录。这样的例子显而易见：应用程序或者系统密码绝不应该出现在日志中（遗憾的是，这有时仍然发生在 Web 应用程序中）。同样明显的是，健康信息本身应该排除在日志之外。

我们应该记录哪些事件？选择日志事件的总体主题是什么？

很明显，我们需要知道谁、什么时候、为什么访问健康信息。我们还需要知道谁添加、更改或者删除信息。但是这并不是全部——我们还需要注意谁试图（但是失败了）读取、更改或者删除信息。如果我们无法记录对每部分数据的访问，就需要谨慎地记录对应用程序本身的所有访问。

接下来，我们需要知道谁在系统上执行了处理健康信息的操作，因为这些活动可能影响未来对保健数据的访问。例如，如果有人关闭日志记录，或者在系统中添加了可能启用无限制数据访问的组件，就需要记录。此外，我们需要记录有关健康信息系统的关键事件，因为这些事件可能为未授权访问提供间接证据。

下面的列表提供上述准则的结构化视图：

- 身份认证，授权，访问：
  - 身份认证 / 授权决策，成功和失败——特别是特权身份认证。
  - 记录用户注销，这对于知道何时用户不再有权访问应用程序也很重要。
  - 从一个用户账户切换到另一个。
  - 系统访问、数据访问、应用程序组件访问。
  - 对应用程序的网络访问，包括分布式环境中从一个应用程序组件到另一个组件的远程访问。
- 变更：
  - 系统 / 应用程序变更（尤其是特权变更）。
  - 数据变更（创建和销毁也是变更）。
  - 应用程序和组件安装及更新、删除。
  - 敏感数据更改、添加和删除。
- 可用性问题：
  - 系统、应用程序和应用程序模块 / 组件的启动和关闭。
  - 故障和错误，特别是影响应用程序可用性的错误。
  - 备份成功及失败（影响可用性）。
- “坏事” / 威胁：
  - 无效输入，其他可能的应用程序违规使用。
  - 恶意软件检测事件。

- 中断或者禁用安全控制或者日志记录——成功及失败。
- 日志记录终止事件，和修改或者删除日志的可能企图。
- 已知会影响应用程序的其他安全问题。

虽然为现有的每个保健应用程序建立一个全面的“记录哪些日志”列表是不可能的，但是上述列表可以为相关的应用程序提供一个有用的出发点。它可以转换为你的应用程序日志记录策略，不需要太多额外的工作。

我们应该记录哪些细节？接下来，每个事件应该记录什么数据，记录的详细级别如何？在此我们使用如下的总体主题：

- 涉及谁？
- 发生了什么？
- 发生在哪里？
- 发生在何时？
- 为什么发生？
- 如何发生的？

在上面我们还提到，能够轻松地集中日志，对于跨多个系统或者分布式应用程序的多个组件的分布式日志分析来说是不可或缺的。虽然 syslog 因其简单的 UDP 传输而成为日志集中化之王，但是现代的跨平台应用程序框架要求对日志传输使用发布 / 订阅模式，类似于现代 Windows 版本中所使用的模式。在这种情况下，安全监控工具可以请求订阅特定的日志事件类型——在必要时以近乎实时的方式接收所有相关日志。

除了“必须记录对敏感保健数据的访问”这一基本结论之外，我们必须提醒读者，随着应用程序复杂度的增长，日志记录的重要性只会越来越高。特别是，分析应用程序行为和分布式及基于云的应用程序之间敏感信息的移动，要求我们将应用程序日志记录置于控制之下。

软件架构师和实施者必须“获取”日志记录——没有其他手段，因为来自网络设备和操作系统的基础设施日志记录不能用于检测和调查对 EPHI 的应用程序级威胁。安全团队（最终负责日志审核）需要指导开发人员建立实用和高效的日志记录，用于监控和调查性活动。

当然，MITRE CEE（cee.mitre.org）之类的日志记录标准有所帮助——但是开发它们并普遍应用可能需要几年的时间。在全球性标准没有出台之前，组织应该快速构建并使用自己的应用程序日志记录标准。HIPAA 依从性为创建和采用这样的日志记录标准提供了很好的动机（来源：http://www.prismmicrosys.com/EventSourceNewsletters-July10.php）。

19.5 FISMA

2002 年的《The Federal Information Seaurity Management》（FISMA）“要求每个联邦机构开发、登记文档和实施机构范围内的计划，为支持机构运营及资产的信息和信息系统提供安全性，包括由其他机构、承包商或者其他来源提供或者管理的信息。”

有些人指责 FISMA“只有文档没有行动”，但是这项法案强调每个联邦机构开发、登记文档和实施组织范围内的计划，加固支持其运营和资产的信息系统的安全。

FISMA 本身没有规定任何日志记录、日志管理或者安全监控，因为它停留在策略、规划和联邦系统风险的级别上。根据这项法案，NIST 已经开发了详细的指南，涵盖 FISMA 依从性的具体细节。例如，http://csrc.nist.gov/groups/SMA/fisma/overview.html 页面介绍如何规划联邦机构的 FISMA 项目。除了 NIST，OMB 收集机构的依从性报告——FISMA 定期验证体制。

详细指南的主要来源是 NIST Special Publication 800-53“ Recommended Security Controls for Federal Information system”，现在是第 3 次修订的版本（http://csrc.nist.gov/publications/nistpubs/800-53-Rev3/sp800-53-rev3-final_updated-errata_05-01-2010.pdf）。该文档描述了日志管理控制，包括审计记录的生成、审核、保护和留存，以及出现审计故障时采取的步骤（来源：http://www.prismmicrosys.com/EventSourceNewsletters-Aug10.php）。

下面我们详细地研究这些指南。

NIST 800-53 日志记录指南

“审计和可审核性策略及规程”（AU 控制）小节专注于 AU-1“审计及可审核性策略及规程”，包含“正式、记入文档的规程，便于审计和可审核性策略以及相关审计和可审核性控制的实施。”这确实是从日志记录策略和日志收集及审核规程开始，实现审计日志记录的正确途径。虽然 FISMA 中的审计控制超出了日志记录的范畴，但是上述指导原则对于日志管理也是非常正确的。

AU-2“可审计事件”引用该系列下一部分中包含的 NIST 800-92。正像预期的那样，需要考虑其他组织单元的风险评估和日志记录，才能创建可审计事件的列表。只在“特殊情况”下审计的事件（如事故之后）也在这里定义。

按照逻辑的顺序，在审计事件列表确定之后，AU-3“审计记录内容”说明了每个事件的详细程度，提供了诸如“事件戳、源和目标地址、用户 / 进程标识符、事件描述、成功 / 失败的指示、涉及的文件名和调用的访问控制或者流控制规则”等应该在每个好的日志记录中出现的例子。对高质量日志记录的讨论可以参考 CEE 标准作品（http://cee.mitre.org）。

AU-4“审计存储容量”和 AU-11“审计记录存储”涵盖了许多组织中的关键主题——日志留存。和 PCI DSS 不同，NIST 指南只提供了选择正确方法的提示，而不是简单的答案（如 PCI 中的 1 年）。

AU-5“审计处理故障的响应”规定了日志记录和日志分析中很重要但常被忽视的一个方面——在日志记录失效时必须采取行动。需要采取行动的例子包括“软件 / 硬件错误，审计捕获机制中的故障，以及达到或者超过审计存储容量”以及其他影响日志记录的问题。

AU-6“审计审核、分析和报告”与所收集的日志数据上发生的事情相关。确切地说，它规定组织以“组织定义的频率”，“审核和分析信息系统审计记录，找出不正确或者不寻常活动的迹象”。同样，NIST/FISMA 没有给出简单的答案（像 PCI DSS 中的每天日志审核）。

AU-7“审计归纳和报告生成”处理报告和总结，这是审核日志数据的最常用方法。

AU-8“时间戳”和 AU-9“审计信息保护”、AU-10“防抵赖”处理用于调查和监控目的的日志可靠性。日志必须精确定时，以避免更改的方式保存。文中提到的选择之一是“硬件实现的一次性写入媒介”。另一个提及的方法是加密的使用。

AU-12“审计生成”本质上确保组织“按照 AU-2 中定义的审计事件列表和 AU-3 中定义的内容，生成审计记录。”

接下来，日志记录指南结束，进入安全监控部分：AU-13“监控信息泄露”的重点是信息窃取（敏感数据的“逸出”），AU-14“会话审计”涵盖用户活动分析的记录（“会话数据”）。我们常说，如果没有异常处理和响应规程，日志记录就是无效的。

总体上说，这里介绍的是成功的 FISMA 驱动日志管理实施可能需要的条件。满足以上要求的方式在同类组织中可能各不相同，就像所有日志管理项目一样。（来源：http://www.prismmicrosys.com/EventSourceNewsletters-Aug10.php）。

你实际上应该做什么？下面将 FISMA/NIST 指南提炼为可以实施和维护的行动项，可以用于需要或强制进行 FISMA 依从性的组织：

- 首先要有日志记录策略。但是如果没有作为基础的操作规程并投入实践，这就没有任何意义（AU-1）。
- 可能需要对多种系统进行配置变更；按照顺序，在文档的其他地方规定配置标准的更新。
- 根据策略，定义所有记录的事件（AU-2）和每个事件所生成的细节（AU-3）。按照 AU-12 启动日志记录。
- 考虑记录所有出站连接性，以检测数据泄露（按照 AU-13），确保记录用户访问会话（AU-14）。
- 定义日志存储方法和留存事件（AU-4 和 AU-11），留存生成的日志。
- 保护日志免遭篡改，保持精确的时间，以维持日志的证据效力（AU-8-10）。
- 根据策略，还要实施日志审核规程和报告生成（AU-6，AU-7）。将报告分发给应该了解这些信息的各方（也是按照第 1 项中创建的策略）。

这时，你的组织应该在策略级别和技术级别上为 FISMA 依从性做好了准备。现在，维护长期的意识完全取决于你。有些组织犯下了危险的错误，停留在策略和规程的水平上，而从未配置实际系统进行日志记录。记住，文档不能阻止恶意的黑客，当日志数据丢失时策略无助于事故调查，而谈论“符合依从性战略”并不能带来安全，甚至依从性。

FISMA 本身没有规定任何日志记录、日志管理或者安全监控，因为它停留在策略、规划和联邦系统面临的风险这一层面上。按照这一法案，详细的指南由 NIST 开发，涵盖 FISMA 依从性的各个细节。详细指南的主要来源是 NIST Special Publication 800-53《Recommended Security Controls for Federal Information system》，现行的是第 3 次修订版，我们在前一个问题中已经介绍过。该文档描述了日志管理控制，包括审计记录的生成、审核、保护和留存，以及审计故障时采取的行动。以此为基础，NIST 创建了专门的“Guide to Computer

Security Log Management”。该指南指出“实施如下建议，应该能够帮助联邦部门和机构实现更高效的日志管理。”（来源：http://www.prismmicrosys.com/EventSourceNewsletters-Aug10.php）为 FISMA 开发的指南还有很多。特别是，美国系统网络安全学会（SANS）已经创建了《Twenty Critical Security Controls for Effective Cyber Defense: Consensus Audit Guidelines》http://www.sans.org/critical-securitycontrols/）。最重要的控制之一“关键控制 6：审计日志的维护、监控和分析”与日志记录和日志管理相关。它映射到 NIST 800-53 AU 和其他控制，特别是我们前面已经介绍的“AC-17 (1), AC-19, AU-2 (4), AU-3 (1, 2), AU-4, AU-5, AU-6 (a, 1, 5), AU-8, AU-9 (1, 2), AU-12 (2), SI-4 (8)”。和 NIST 800-92 中日志记录的端到端覆盖（这种方式可能使随意的读者不知所措）不同，SANS 文档包含了“速效方案”，机构可以立即按照这些方案开展 FISMA 工作。

如何在组织中使用 NIST 800-92 文档？首先，我们要熟悉一下这 80 页文档中有什么内容。该指南从对计算机安全日志管理的简介开始，然后是 3 个主要的部分：

- 日志管理基础设施
- 日志管理规划
- 日志管理运营过程

确实，这是考虑任何日志管理项目的正确途径，因为组织通常在规划、日志记录基础设施建设上遇到挑战，然后是持续的运营，这必须在组织存在的整个生命期中得到维护。

该指南将日志管理定义为“生成、传输、存储、分析和处理计算机安全日志数据的过程”。顺便说一句，要记住，安全日志管理必须涵盖“来自安全应用程序的日志”（如 IPS 警报）和“来自应用程序的安全日志”（例如业务应用程序中的用户身份认证决策）。只重视其中一个是错误的。

在日志管理基础设施的领域中，该指南定义了日志管理基础设施的三个层次：

- 日志生成。
- 日志分析和存储。
- 日志监控。

遵循上述顺序是日志的常识；但是很遗憾，许多组织有时从购买昂贵的工具开始，而没有考虑日志记录策略和日志的使用。在考虑逻辑单元和软件之前，先考虑需求（你想从日志中得到什么？）和日志（哪些日志能够帮助我实现目标？），能够帮助你的组织避免许多头疼的问题。

日志管理项目规划从关注一个关键项目——组织角色开始。日志管理本质上是“水平”进行的，涉及组织的许多领域。NIST 建议，邀请系统和网络管理员、安全管理员、事故响应人员、CSO、审计人员甚至内部应用程序开发人员加入。这能帮助组织选择并实现对日志管理问题的正确回答（来源：http://www.prismmicrosys.com/EventSource-Newsletters-Aug10.php）。

接下来，真正的工作开始了：创建日志记录策略。“安全始于策略”是一个公共的主题；

这非常适合于日志记录。根据指南，需要涵盖如下策略：

- 日志生成：记录哪些事件，包含哪些细节。
- 日志传输：日志如何收集，如何集中整个环境的日志。
- 日志存储和处理：留存和事后处理日志的方法和位置。
- 日志分析：如何解读日志事件，如何据此采取行动。

这在实践中意味着什么？它意味着创建了涵盖所要完成工作的策略之后，才需要配置工具。首先是目标，然后才是基础设施的选择！在以 FISMA 为基础的隐私和其他法规案例中，法律部门也应该参与，但是这可能使安全团队感到不快。

在定义策略和构建实施策略的系统，并配置日志来源之后，构建长期持续计划的艰苦工作开始了。这种计划的核心是对日志数据进行定期分析，并对识别出来的异常情况采取相应的措施。显然，任何外部指南都不能定义组织中什么是最重要的，但是参考本书、NIST 和其他指南，你对于最应该关心的日志已经有了一些想法（来源：http://www.prismmicrosys.com/EventSourceNewsletters-Sep10.php）。

机构在执行与日志数据长期管理的任务时，频率较低一些。如果你的日志容量达到 TB 级别或者更多，这是一个很大的问题。NIST 800-92 建议首先选择“用于数据存档的日志格式”——原始、解析等。和 PCI DSS 一样，NIST 文档也包含了安全存储日志数据和完整性验证的指南。

首先，它为日志管理计划打下坚实的基础，许多其他强制法规专注于工具，而 NIST 文档包含了非常有用的计划管理技巧，归纳了避免日志分析人员被太多的日志弄得筋疲力尽的方法。

其次，你可以使用这一指南，学习日志管理中常被忽视的方面：日志保护、存储管理等。例如，它包含了排定日志记录审核优先级的一些实用技巧。

第三，它为你提供了证明日志管理领域决策的方法，即使你不为政府机构工作，这些方法一样适合。

与此同时，该指南更多的是关于过程，而不是关于“位”和“字节”这些细节的。它不会告诉你哪个工具最适合。

实际上，尽管 NIST 800-92 不是与联邦政府之外的组织相关的指南，但是商业组织也可以从中获益。例如，即使是以 PCI DSS 作为主要目标的零售组织，也可以根据 800-92 构建其日志管理计划。他们使用 NIST 指南进行工具选择，作为策略模板的来源，甚至确保日志管理项目的持续成功运营。

为遵循 FISMA 的机构编写的其他技术日志管理指南由 SANS 以其《Twenty Critical Security Controls for Effective Cyber Defense》(CSC 2.0）的形式发表。如果你寻求的是快速可行方案（文档中称作“Quick Wins”），这一资源就适合你。例如：

- 对每个硬件设备和其上安装的软件，验证审计日志设置，确保每个数据包或者事务的日志包含日期、时间戳、源地址、目标地址和各种其他有用元素。系统应该以标准格

式记录日志，如 syslog 条目或者通用事件表达式（CEE）倡议所定义的格式。

- 系统管理员和安全人员应该设计指定系统中常见事件的简档，以便调整检测，专注于不寻常的活动……
- 对内部网络的所有远程访问，不管是通过 VPN、拨号线路还是其他机制，都应该有详细的日志。

建议使用 NIST 800-53、NIST 800-92 和 SANS CSC2.0，优化 FISMA 依从性项目的日志记录。

总结起来，NIST 800-92 和 SANS CSC2.0 教会我们进行如下的工作，不管是仅为 FISMA 依从性，还是多重法规计划或者改进安全性和运营：

- 寻找必须记录日志的关键系统。
- 启用日志记录，并确保日志满足标准中提到的“好日志”的准则。
- 在日志记录活动中包含不同团队，日志记录横跨整个机构。
- 查看你的日志！从现在开始而不是等到明天。
- 尽可能自动化日志管理，在所有领域都指定可重复的稳定过程。（来源：http://www.prismmicrosys.com/EventSource-Newsletters-Sep10.php）。

以此为基础，NIST 800-92 将日志管理带进有着“日志？让它们烂掉吧”想法的人们的视线之内。它的过程指南要比技术指南更广泛，对 IT 管理非常有用，而不仅仅适合于“壕沟里的人”，对于那些人来说，他们可能已经知道日志中有宝贵的东西…

19.6 小结

总结起来，日志记录和监管依从性联系紧密，在未来仍然是如此。尽管日志有很多挑战，但是日志记录仍然是 IT 可审核性的主要手段，因为大部分用户和系统操作可能在日志中记录。这也就是日志记录成为完善的依从性技术，被许多法规和法律所强制的原因。

Chapter 20

第 20 章

规划自己的日志分析系统

本章要点

- 规划
- 软件选择
- 策略定义
- 架构
- 扩展

20.1 概述

本书已经接近尾声，现在是提供实践指南的时候了。在本章中，我们将带你熟悉一个检查列表，可用于日志分析系统部署的规划或者考虑。如果你打算购买一个商业化的系统，本章中介绍的这些内容一般由你所选择的供应商处理。但是，你必须参与项目中的每个步骤。最终，你必须让供应商满足你的需求，没有别的路可走。如果你发现自己需要部署开源解决方案，本章也能很好地帮助你。应该注意的是，哪些小节适合于你，取决于你在过程中所处的位置。例如，如果你已经确定了软件，但是对于硬件的分配没有头绪，可以忽略 20.3 节“软件选择”的部分。不管你选择那条途径，在许多情况下都可以使用本章提供的指南。

20.2 规划

除了确定产品或者服务之外，你必须要做的第一件事是理解对这样的产品的具体需求。

你应该考虑的因素包括：

- 用例：你能够定义精确的场景，说明系统是如何使用的吗？例如，下面是一个用例的样板："作为系统的用户，我想要运行一个报告，告诉我过去 24 个小时 Windows 服务器上的登录失败次数。"
- 驱动因素：是什么"驱动"你启动这一项目？是因为有趣、必要性还是政府法规或者其他目的？如果你能够回答这个问题，就能为驱动因素指定一个优先级（例如，如果只是因为有趣，优先级应为"低"），这有助于你清晰地了解为此分配的时间、金钱和资源。
- 解决的问题：部署一个日志分析系统能帮助你解决其他问题吗？例如，由于你收集的信息类型，日志分析系统能够帮助你进行服务器或者网络监控。
- 安全和依从性：这样的产品是否帮助你向审计人员、监管人员证明依从性？

好了，现在项目已经启动，你必须开始规划。如果你有自己的项目管理团队，你可能希望他们帮助你管理和跟踪项目。如果你一切自己动手，那么就将身兼两职。

20.2.1　角色和职责

你首先必须确定角色和职责，原因之一是这能够帮助你确定，在项目规划和实施阶段，应该包含哪些人。而第二个原因是这项工作帮助你了解，组织中的哪些人可能是日志分析系统的用户。例如，程序员是日志记录系统的用户，因为他们编写的应用程序必须创建日志消息，用于诊断和故障排除目的。

表 20.1 提供了常见角色和职责的一个摘要。

表 20.1　角色和职责摘要

角　色	职　责
日志分析系统管理员	这个角色可能由你承担。作为这一角色，你负责系统的维护和保养。这包括安装软件更新，配置日志收集器和其他任务。另一个任务是与组织的其他成员一起工作，理解他们的对分析系统的需求。换言之，如何帮助其他人从系统中得到价值
业务 / 产品负责人	对产品、特性集、应用程序负有责任的个人或者群体。这一群体通常对日志记录有自己的需求，但是他们可能不知道（或者不需要知道）有一个分析系统。帮助其他人了解自身需求的责任可能落在你的身上
安全团队	安全团队通常负责配置安全设备上的日志记录，确保组织的安全态势
IT/ 开发经理	将日志记录标准作为影响整个组织的"必备"功能
IT 支持团队 / 管理员 / 网络操作人员	该团队支持生产应用程序；他们可能有在应用程序表现异常时读取和分析日志文件的用例
软件开发人员 / 程序员	通过使用库、API 和日志记录功能，遵循日志记录标准。在开发人员需要跟踪应用程序问题等时候，日志分析系统对他们也很有用
事故响应团队	事故响应团队被视为日志分析系统的客户。他们将在事故响应和升级期间使用收集的日志
内部 / 外部审计人员	根据所在企业的业务类型，你可能必须应对审计人员。审计人员通常需要验证你正在收集某类日志，以便确定和各种法规的依从性

20.2.2 资源

资源和理解组织关于日志记录的需求，以及日志记录对环境的影响有关。下面是在规划阶段你可以向自己提出的问题。

1）你的现有员工中，哪些人能够进行日志分析和监控？

2）你的现有员工中，哪些人能够进行日志分析系统的管理？

3）你是 24×7 运营、只在规定时间内运营，还是采用混合模式？

4）你是否需要构建一个安全运营中心（SOC），或者拥有一个虚拟 SOC？虚拟 SOC 团队不是不间断地坐在那里盯着屏幕，而是使用警报来驱动响应、工作流等。

5）你对事故响应有何计划？

6）你在日志分析系统的构建上有何种预算？

7）你在长期运行和维护系统（硬件升级、网络升级、人员升级等）上有何种预算？

对这些问题的回答有助于更好地理解项目和最终实施所需要的资源。例如，如果你计划建立 24×7 的运营中心，需要考虑如下因素：

- 你需要雇佣新员工，还是使用内部员工？
- 你的办公室里是否有场所可以安置 24×7 运营中心？
- 你的电力容量是否足以供应 24×7 中心？
- 你是否有预算购买运营所需的硬件和软件？
- 你的事故响应和升级规程是怎么样的？
- 你负责生成哪些类型报告（例如交给管理层的报告）？

有所了解了，对吗？上述列表还可以延长下去。

最后几个需要考虑的因素是与日志分析系统部署有关的软硬成本。硬成本通常与资本性支出相关，如运行系统的硬件、支持系统的基础设施升级、操作系统成本等。软成本集中在部署、维护、监控所需要的时间。这些活动似乎不需要什么成本，实际上却并非如此。

很明显，在承担规划日志分析系统实施之类的任务时，你有很多需要思考的问题。

20.2.3 目标

在项目之前、之中和之后，设置目标都很关键。目标帮助你知道自己的进展。有些目标可能是标志着项目可交付成果的里程碑。下面是一组日志分析系统规划中的目标和里程碑。

- 目标 1——规划之前
- 目标 2——利益相关方确认
- 目标 3——需求收集
- 里程碑 1——项目规划创建
- 目标 4——日志分析工具评估
- 目标 5——日志分析工具选择
- 目标 6——硬件需求收集

- 里程碑 2——硬件和软件采购
- 目标 7——软件部署
- 目标 8——构建运营中心
- 目标 9——为运营中心配备人员
- 目标 10——定义事故响应和升级规程
- 里程碑 3——系统投产

20.2.4　选择日志记录的系统和设备

如果你没有设备开启日志记录，日志分析系统就不能算是好的。规划过程中一个关键的方面是确定用于评估和选择日志记录的系统和设备的可重复性过程。为设备指定一个关键程度是其中一种方法。例如，防火墙的关键程度高于工作站。下面是选择日志数据源过程的一个示例：

1）关键性：你必须理解日志数据源对环境的关键性。它是企业中主要的 Web 服务器吗？它是包含公司机密的关键研究与开发服务器吗？遭到入侵就会威胁你的生存的系统应该排名靠前。不同的日志源也可能对你的企业有不同的关键性。例如，你的 DNS 服务器关键性可能低于信贷处理服务器。

2）验证：一旦识别了数据源，该系统有可能归另一个小组所有（有时这个小组由你负责）。你必须召集利益相关方，验证这个数据源确实是关键资产，需要纳入日志分析系统。

3）事故响应：最后，你需要围绕指定日志数据源遭到入侵时，事故处理和响应如何进行，确立一个工作流。发生事故时，你必须致电谁？在遭到入侵的主服务器上进行调查时，该系统有没有热备份供你进行故障切换？

20.3　软件选择

选择日志分析系统时，对于开源、商业化或者托管服务有不同的考虑。本节简单地概述研究日志分析解决方案时，最应该考虑的因素。

20.3.1　开源软件

日志分析的开源选项多种多样。在第 15 章中已经列举了一些。本节将提供研究这些工具时更重要的一些考虑因素。

1）更新频率：社区和核心开发团队对该项目是否经常投入？如果你在几个月或者更长的时间没有看到活动，这可能是一个衰亡的项目。

2）成本：有人会说，开源意味着“免费”。这并不总是对的，开源项目可能以多种形式给你造成成本。你一般需要花费更多的精力去获得、学习和部署开源解决方案。

3）许可证：开源应用程序与何种许可证发行往往值得注意。它所使用的许可证可能限

制部署软件的方式，甚至规定你的定制软件和该工具集成的方式。

4）支持：许多开源工具提供你可以购买的支持合同，包括电子邮件支持和其他特性。有些工具甚至在工具的“免费”版本（有时称作社区版本）之外，提供增强的付费版本。增强版本通常有某种程度的支持。

5）功能：确保你完全理解该工具能做和不能做的事情。这是你需要确实了解自身需求的时候。如果你首先定义需求，就能更好地评估日志分析工具。

6）文档：这是显而易见的考虑因素。如果应用程序只有一个 README 文本文件作为文档，就可能不是一个好的方案。

7）社区：这有时候是关键的特性。你肯定希望了解开源工具是否有某种社区的支持。这种社区通常以留言板的形式提供，你可以在那里获得该应用程序的帮助和支持。

8）可扩展性：如果你希望扩展日志分析系统（可能采用自定义脚本、程序等），就可能想要了解平台的可扩展性。它有没有 API 和其他集成点？如果平台支持这种扩展，在出现问题时有谁来支持？

9）功能：需要考虑和研究的功能包括：

a）支持的日志获取方法（syslog、原始文件、Checkpoint 的 OPSEC API 等）

b）是否易于维护和保养（例如，是否容易添加新设备源的支持？）

c）关联（是否支持基于规则、统计等）

d）报告（有哪些现成的报告，是否容易创建新的报告？）

20.3.2 商业化软件

在软件业的许多不同的细分市场中，商业化软件总是与企业级解决方案联系在一起。这种关联常常是正确的，但是在日志记录和日志分析中，“细节是魔鬼”。下面我们来看看，在考虑商业化日志分析系统时，需要知道的几件事：

1）成本：这是显而易见的考虑因素。商业化日志分析系统根据网络的大小、计划留存的数据等而各不相同。

2）支持模式：商业化解决方案的支持是确保你从平台中获得最大价值的关键。你必须了解支持模式的各个方面，了解它与你的业务需求（如正常运行时间、周转时间等）是否保持一致。

3）专业服务：大部分商业化供应商提供某种专业服务，帮助你安装和设置软件。这通常需要额外付费，换言之，它不是免费的。

4）功能：在 20.3.1 节中提到的项目也同样适合于商业化软件。

5）可扩展性：和开源软件类似，你应该感受产品的可扩展性。

托管安全性服务提供商（MSSP）

MSSP 为构建和部署自己的日志分析提供了一个替代品。寻求 MSSP 和本章的性质并不

完全一致，但是它是值得一提的替代方法，其吸引力主要来自组织不需要担心人员的招聘和培训，软硬件的部署和维护以及其他琐事。MSSP（尤其是顶级的）的明显好处之一是，它们有训练有素、经过认证的分析人员。这对于没有同样专业性的客户来说是巨大的优势。研究 MSSP 时需要考虑的一些因素如下：

1）功能：需要研究的 MSSP 功能不同于现场日志分析系统。以前提到的大部分功能都由 MSSP 维护。更应该关心的功能包括客户门户的实用性。该门户是你用于搜索日志消息、运行报告、响应事故和工作单等的主要界面。

2）分析和事故响应：需要考虑的因素之一是 MSSP 如何进行分析和事故响应。这对于你的业务可能很关键，必须确保 MSSP 与你的目标相一致。

3）专业服务：MSSP 提供的专业服务可能有用，特别是需要现场事故响应的时候。这些服务通常需要附加成本，超出了常规服务的费用。

4）数据留存：确保你理解日志数据留存的时间。你可能必须符合与留存相关的某些监管原则。你可能需要至少一年，甚至更久。

5）云日志：第 21 章详细介绍云日志的主题。基本要点是，提供者在其数据中心提供一个环境，你可以将日志转发给它们。这种方法不需要在现场有特殊的硬件。这对于某些组织可能很理想，但是对其他组织就不一定了。不同的依从性和法规可能禁止某些组织以这种方式运营。

6）服务水平协议：MSSP 提供服务水平协议（Service Level Agreements，SLA）。在 MSSP 的环境中，SLA 规定了分析、事故响应和升级的低限和高限，通常在你签署的合同中列明。

7）使用数据以保护数据：因为 MSSP 监控大量客户的安全性，在某个客户检测到威胁和入侵时，创建保护并应用到所有客户。这是所有客户都能获益的强大工具。

20.4　策略定义

策略定义处理一组规程的创建，你将把这些规程作为例行程序的一部分。本节将简单描述较为常用和必要的策略，需要在规划日志分析系统时加以考虑。定期和经常审核你的策略，以确保它们与组织的需求和需要一致，也是很关键的。

20.4.1　日志记录策略

最直接的关注点是为组织定义一个日志记录策略。日志记录策略包括：

- 覆盖日志事件类型（登录 / 注销、资源访问、防火墙接受 / 拒绝、IPS/IDS 警报等）和细节的足够日志记录。
- 日志聚合和留存（1 年等）。
- 日志保护（确保日志不被篡改）。

- 日志审核

日志记录策略定义日志数据应该捕捉哪些属性供以后审核、升级和响应。日志记录策略还扩展到应用程序。例如，如果你有定制的应用程序，有必要和应用程序利益相关方一起定义哪些信息应该记录，并使之可用于日志记录系统。

20.4.2 日志文件轮转

日志文件轮转是另一个需要定义的策略。随着你所监控的日志文件的增长，本地系统上的磁盘会被充满。磁盘满时，应用程序可能无法正常工作。更糟糕的是，你可能丢失关键的日志数据。所以，按照如下方法，开发一个全组织的日志文件轮转策略：

- 基于时间：根据时间范围轮转日志文件，也就是说，按照分钟、小时等。
- 最大化实用性：这里的关键是使系统对用户有用。这需要采取平衡的措施：轮转的日志需要保留足够长的时间以确保其实用性，同时又不能影响整个系统。
- 及时转移到长期存储：这里的思路是，日志保留在源系统上的时间长度要足以用于审核。根据系统的繁忙程度，这一时间大约是几个小时。

20.4.3 日志数据收集

数据收集是日志分析系统中最重要的部分。没有这一功能，你就无法用日志分析系统实现目标。你首先应该了解环境中哪些设备应该配置为生成日志数据。例如，你的激光打印机能够生成日志数据，并不意味着你打算捕获它。日志数据收集的策略需要根据环境进行调整，例如，你可能只需从外部防火墙和 IDS/IPS 系统中获取日志消息。你也可能需要收集来自网络中每个防火墙、IPS、服务器和桌面电脑的日志记录。无论如何，你应该确保每 3 至 6 个月审核这个策略，确保组织的需求与你的数据收集策略相符。

20.4.4 留存 / 存储

日志留存通常是许多法规的要求。回忆一下第 4 章，日志留存实际上由三部分组成：日志存储、可访问性和日志销毁。绝不要用 syslog 服务器作为日志留存系统，这是必须禁止的。还要注意的是如何确定留存所需的存储容量，下面是一个简单的算法：

每秒日志记录数（以字节计算）× 86400

这一计算会告诉你每天需要的存储量（字节数）。难点在于了解整个环境中每秒生成的日志记录数量。这可能需要在分析系统投产和运行之后，进行一些早期的发现和试验。你通常可以在配置 / 调整模式下运行系统，这时系统会从日志源获取数据，但是不对数据采取任何行动。这样你就有时间确定设备和系统的配置是否合适。在这一阶段，你可以判断环境的日志数据负载。通用的经验法则之一是在所需的日志留存容量上增加 25%，用于规划不可预见的事件。

记住，你并不一定要采用大型服务器和磁盘阵列。你所选择的日志分析解决方案可能支持分布式存储模型，在这种模型下，接收日志数据的系统节点也可以作为留存点。在评估日志分析系统的时候，记得考虑这一点。

20.4.5　响应

第 16 章提供了事故响应和升级的许多细节。我们在此提及，是为了让读者注意到，它也是策略方案中重要的考虑因素，对这一主题的更详细介绍参见第 19 章。

20.5　架构

日志分析系统的部署很大程度上取决于网络的规模，以及你从网络的哪些部分收集日志。我们讨论 4 个基本部署模型：

1）基本模型

2）日志服务器和日志收集器

3）日志服务器和具有长期存储的日志收集器

4）分布式部署

20.5.1　基本模型

基本部署模型是你可以使用的最简单模式。它由单个日志服务器组成，多个设备向其发送日志。图 20.1 描述这种情况。

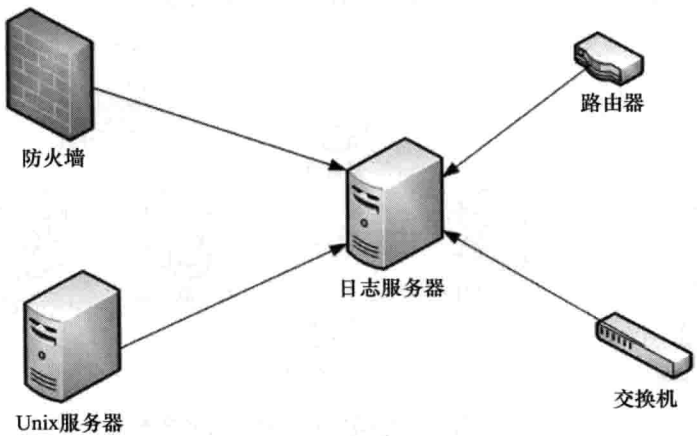


图 20.1　基本日志服务器部署

在这种模型中，你一般将日志记录到日志服务器，以进行日志审核。这不是最理想的情况，但是在某些条件下是你的最佳选择。

20.5.2 日志服务器和日志收集器

较常见的一种部署模式是将日志收集器分布到网络中有战略意义的位置。这些系统从他们所在的设备中收集日志。这些日志收集器可能是网络中的特殊部分（可能是一个入口点），你应该确保捕捉防火墙日志。日志收集器将把日志转发到集中日志服务器。图 20.2 展示了这种部署。

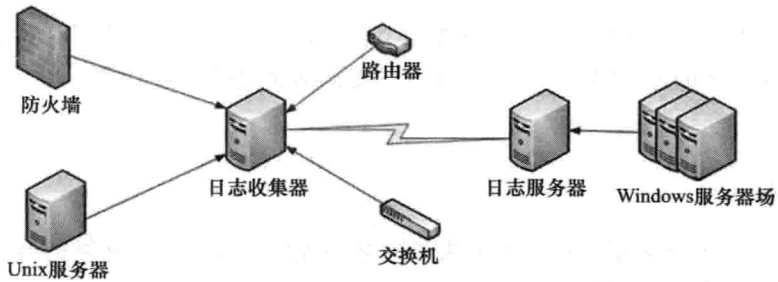


图 20.2　日志服务器和日志收集器部署

图中，我们让日志收集器向日志服务器发送日志。还要注意，日志服务器本身也可以作为日志收集器（注意图中的 Windows 服务器场）。根据你所选择的日志分析系统，这种部署的配置细节将是特定于供应商的。日志服务器作为审核日志、分析日志等工作的集中位置。日志收集器可以看作日志的临时备份，但是绝不能依靠它。

20.5.3 日志服务器和具备长期存储的日志收集器

在这个模型中，我们简单地添加长期存储。图 20.3 显示了这种模型。

日志服务器写入某种长期存储。同样，配置是特定于供应商的。

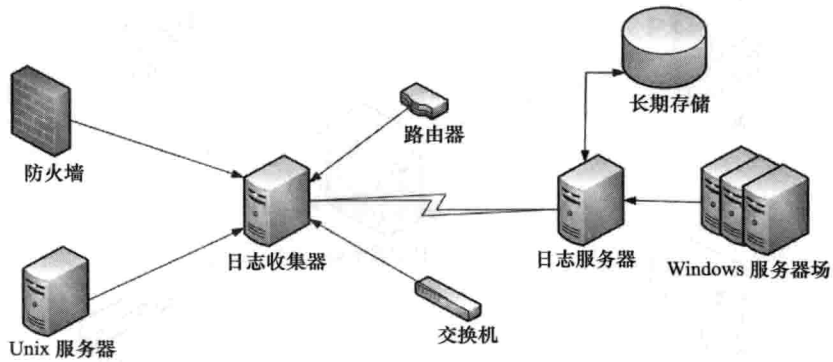


图 20.3　具有长期存储的基本日志服务器部署

20.5.4 分布式部署

分布模型我们实际上已经讨论过了。你使用分布在整个环境中的日志收集器收集日志。

这些日志收集器将日志汇聚到一个或者多个日志服务器。在层次化模型中，你可以让多个日志服务器将日志积累到一个集中日志服务器，提供环境的高层视图。这也是在日志服务器级别进行自动化分析的情况，考虑了分布式分析方案，可以减少单一服务器完成这种任务的需求，减少了服务器负载并缓解了单一故障点。

20.6 扩展

扩展是日志分析系统管理的重要部分。下面是用于协助这种努力的一些通用技巧。

1）了解你的环境。为了扩展，你需要掌握进出网络的设备。确保你和利益相关方举行每周会议，审核部署新设备的计划。确保你的分析系统中包含合适的设备日志记录，对于最大化这种部署的实用性是至关重要的。

2）确定你知道网络中的哪些日志数据更为优先。例如，防火墙日志可能比来自打印机的日志更重要。

3）理解环境何时饱和。在日志审核中，如果你注意到日志数据是几个小时前的，就可能碰到了问题。如果你采用分布式部署，应该调查日志收集器和日志服务器，看看为什么它们落在后面。为什么服务器有很高的 CPU 负载？它遇到了 I/O 问题吗？这些情况都是数据容量增加的信号。这可能是恶意活动的指示，也可能你需要引入新的日志收集器，将日志数据转移到新的服务器。

20.7 小结

本章为你提供规划和部署日志分析系统时所需要的指导。我们介绍了规划、策略定义、软件选择、部署和扩展。这些都是理解选择和部署一个日志分析系统的方法和原因的重要部分。

Chapter 21 第 21 章

云 日 志

本章要点

- 云计算
- 云日志
- 监管、依从性和安全问题
- 云中的大数据
- 云中的 SIEM
- 云日志的优缺点
- 云日志提供者目录
- 其他资源

21.1 概述

本章描述围绕云计算、云安全以及云日志的基本概念（本章的重点是云日志）。本章的目标不是让你成为这些主题的专家，而是为你提供基本知识，以及做出云日志方面的明智决策所需的理解。为什么这一点很重要？你可能认为自己需要考虑为日志记录、安全、安全信息和事件管理（SIEM）或者其他软件应用程序及系统采用基于云的提供者。你对云概念的细节了解得越多，就越容易在选择提供者时做出正确的决定。

21.2 云计算

什么是云计算？云计算已经成为近几年许多报道的主题，因此对于云计算的描述不在少数。本质上，云计算是一个用于描述用户与一组计算机资源（CPU、内存、磁盘等）交互方式的术语。对于最终用户，他好像在使用一个计算机系统。但是，实际上用户使用的可能是许多不同（往往进行虚拟化）的组件。用于说明云计算的一个例子是电力公司。我们的家里都有电源插座，这些插座可以看作简单的接口，当我们希望使用烤面包机、计算机、电脑等设备时，只要将设备插入墙座就可以了。我们并不关心如何发电，所知道和关心的是墙座能供电。云计算的关键特征之一是弹性（elasticity）。弹性是扩展（在需要更多资源时）和资源不再需要时收缩的能力。

在本章的其余部分，将使用“提供者”和“消费者”这两个术语。我们有必要简单地说明它们。提供者是提供云服务的一方，可能是第三方实体，也可能是公司内部的一个群体。有时候，也使用云日志提供者的术语，这更明确地表示我们在本章中讨论的一类提供者。消费者是消费云服务的人，两者的区别很快就能分清。

21.2.1 服务交付模型

为了进一步说明云计算，有必要了解服务交付模型。服务交付模型指的是特定的计算机资源交付给最终用户的方式。在下面几个小节里，我们将讨论 SPI——软件即服务（Software as a Service）、平台即服务（Platform as a Service）和基础设施即服务（Infrastructure as a Service）。下面就开始我们的讨论。

1. 软件即服务（SaaS）

SaaS 这个术语用于描述存在于提供者基础设施上的软件。消费者使用瘦客户端或者 Web 浏览器交互并使用该软件。从消费者的角度看，他们是与单一计算机系统交互。他们不负责软件和硬件维护、保养等工作。应该注意的是，SaaS 提供者可能是 Salesforce.com 这样的第三方外部提供者，也可能是消费者所在组织内的内部提供者。云日志提供者属于这种交付模型，也就是说，一个组织接受日志，以便为其他组织分析日志。

2. 平台即服务（PaaS）

在 PaaS 模式下，消费者可以将定制的应用程序或者工具部署到提供者的云。和 SaaS 一样，消费者对底层硬件或者网络没有控制权（在某些情况下，消费者可能访问防火墙配置），但是和 SaaS 不同，消费者对于部署的应用程序和相关配置有控制权。云日志提供者不允许消费者安装自己的应用程序。

3. 基础设施即服务（IaaS）

最后一种服务交付模型处理基础设施。IaaS 允许消费者配置计算资源，如 CPU、内存、

磁盘、网络和其他相关资源。消费者通常控制操作系统、部署的应用程序和存储容量。他们不能控制底层基础设施。

4. 存储即服务

顾名思义，存储即服务处理云中的存储。消费者和提供者签署一个协议，租用提供者网络上的空间。价格通常根据每 GB 存储和某种数据传输率确定。最常见的用例是场外备份和检索服务。中小规模消费者可以从这种服务获得很大的利益，因为 IT、硬件、软件和网络管理都由提供者承担。

21.2.2 云部署模型

我们已经讨论了云的交付模型，下一个需要了解的就是服务部署的方式。“云计算系统可能进行私有部署，或者托管在云客户的场地中，可以和有限数量的可信合作伙伴共享，可以由第三方托管，也可以是公众可以访问的服务，即公共云”(Web，2011b)。你可能问自己，为什么应该关心不同的部署模型？从云日志的角度来看，部署模型的重要性在于，你需要在评估这些提供者时了解这些选项。例如，提供者可能在众多云计算提供者中的某一个基础上构建其服务，也可能构建自己的私有云。

理解部署模型的两个关键概念是场内（On-premise）和场外（Off-premise）。场内指的是基础设施位于消费者的场地——也就是他们的数据中心。你可能已经猜到，场外指的是云处于消费者的场地之外——也就是云提供者的数据中心。下面的几个小节描述常见的部署类型。

1. 公共云

顾名思义，公共云的基础设施可供公众访问。这种云一般是由行业集团或者公司拥有，以各种不同的形式销售云服务。云的基础设施由某个组织运营，可以由该组织或者云提供者管理。基础设施可以在场内，也可能在场外。

2. 私有云

同样，从名称就可以看出，这种云是私有的，本意是供单一组织使用。云的管理由该组织或者云提供者负责。私有云既可以是场内的，也可以是场外的。

3. 社区云

有共同关注点的组织可以采用社区云。共同关注点可以是“任务、安全需求、策略或者依从性考虑”（云安全联盟，Cloud Security Alliance，2009）。云的管理由社区或者云提供者执行。社区云可以是场内的，也可以是场外的。

4. 混合云

混合云考虑以某种方式将两种或者更多的云类型（公共、私有等）组合在一起，提供可

移植性。例如，云日志提供者可能允许消费者创建自己的私有云，以便提供者的应用程序栈可以放在同一个位置。这样做的好处是为消费者提供处理灾难恢复、集中化、依从性等问题的手段。

21.2.3 云基础设施特性

我们必须介绍的下一个云计算主题是云基础设施的特性。我们的思路是揭示云基础设施和传统计算环境的不同之处。

- 按需自助服务：消费者可以配置服务的各个方面（CPU、存储、网络等），而不需要与云提供者交互。这通常使用用户界面和 API 来完成。
- 广泛的网络访问：云基础设施可以通过各种具备网络能力的设备访问（如智能手机、PC、便携电脑、平板等）访问，这一点很关键。
- 资源池化：提供者使用允许物理计算资源池化的基础设施，为许多消费者服务。这隐含了多租户的模型。用于实现的最常见技术是虚拟机。这样做的好处是，消费者不需要知道和关心他们的资源是如何管理的。
- 快速弹性：这一点已经讨论过。基本的思路是资源能够根据消费者需求快速增减，可以人工或者自动完成。
- 可度量服务：云提供者通常计量和度量他们所提供的服务（CPU、内存、存储等）。度量的时间单位通常是小时。你只需要为所使用的服务付费。云日志提供者通常根据你被允许发送到云的日志数据量以及你打算留存日志的时长计费。
- 多租户：由于云的多租户特性，有“为不同消费者构成建立策略驱动实施、分段、隔离、治理、服务水平以及成本分摊 / 记账模型”的需求（云安全联盟，2009）。

关于这些主题的更多信息，可以参见云安全联盟（2009 年 12 月）和 Web（2011a，2011b）。

21.2.4 标准？我们不需要讨厌的标准

本节的最后一部分是关于标准的。正如云安全联盟（2009 年 12 月）所指出，“还要注意，许多专注于开放和专利 API 开发，以寻求云的管理、安全和互操作性的活动正在进行。”为什么拥有开放的 API 对云提供者很重要？考虑一下这种情况：你正在使用云提供者 XYZ，他们提高了使用成本，或者降低了服务质量（QoS），你也可能决定将云中的功能搬回家。有了开放的标准和 API，你就可以从一个提供者迁移到另一个提供者，并且带走你的数据、元数据、配置等。已经在进行的标准化尝试包括：

- 开放云计算接口工作组。
- 亚马逊 EC2 API。
- VMware vCloud API
- RackSpace API
- GoGrid 的 API

注释 云计算提供者

提供云计算提供者的详细列表没有意义，因为在本书付印之后，这些内容将会过时。今天较为常用的提供者是 Dell、Rackspace 和 Amazon。我们建议进行互联网搜索，找到这一领域当前的运营商。

开放的 API 有助于确保云提供者之间的可移植性。但是，当提供者最终知道消费者可以轻松地迁移到其他提供者，这将迫使行业实现最佳实践，追求创新，并且正确地对待消费者。

21.3 云日志

我们最终来到了本章的主题。虽然本章是关于云日志的，但是我们必须提供关于云计算的许多支持信息。这是因为，云日志只是云计算的一个应用。换言之，正是云计算才使云日志得以存在。实际上，一种新的部署模型已经出现：日志即服务（LaaS）。LaaS 指的是提供者用云设置来提供日志记录服务。也就是说，提供者获取日志数据，并代表其他人实施分析。本节将为你介绍云日志的概念，它并不能使你成为专家，而是对这一应用留下印象，了解它对你和你的组织有何好处。

图 21.1 展示了云日志的概要描述。

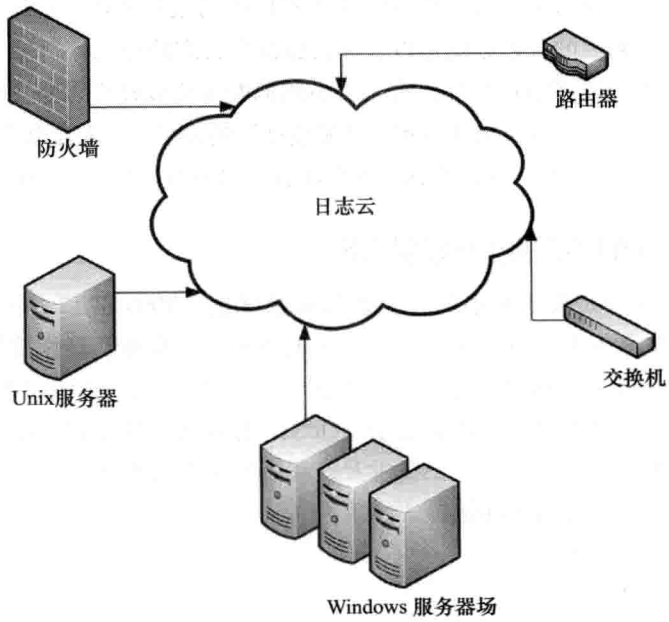


图 21.1　云日志的逻辑布局

图中是一系列将日志记录到云中的设备。但是云里有什么？作为消费者，你不关心这

一点。提供者给你一个 IP 地址或者主机名，让你的 syslog 配置指向它。但是，和往常一样，实际情况更加复杂，至少对于云日志提供者来说是如此。例如 IP 地址的另一端可能是负载平衡器，用于处理入站连接。平衡器把请求发给许多机器中的一个，这些机器获取数据、规范化（并不总是如此）、索引，并将日志存储起来供以后检索。

云日志是一个相当新颖的领域，提供者的复杂度和功能正在不断增加。下面的列表介绍了云日志实体共有的特性和功能：

- 日志获取方法：为了将数据输入到云中，大部分云日志提供者支持 syslog（TCP/UDP 和安全 /TLS）和一个专有的 API（通常是通过 HTTP/HTTPS 的 REST 风格 API）。syslog 覆盖了世界上 95% 以上的设备。API 允许将云日志构建到你的应用程序中。
- 部署模型：有些云日志提供者在云计算提供者的基础上构建他们的服务，而其他一些提供者构建自己的云。提供者甚至可能支持混合模型。
- 日志留存：所有提供者都支持日志数据的长期留存，不同之处在于存储的收费。通常，你控制保留数据的周期（几天、几周、几个月）。有些提供者只允许最多留存 6 个月的数据。
- 核心功能：日志云服务的基本核心功能包括日志数据获取、日志数据索引（用于快速搜索等）、长期存储和用于搜索及审核数据的用户界面（通常基于 Web）。除了这些，还有关联、定制警报和定制报告等额外功能（有些提供者还不支持这些功能）。
- 记账：大部分云日志供应商支持随用随付模型。你不需要签订合同，只需为发送给供应商和希望留存的日志数据付费。而且，大部分提供者有“购买前试用”计划，这种计划本质上允许你每天免费发送少量日志消息，以尝试服务是否令你满意。
- 灾难恢复（DR）：有些云日志提供者在服务条款中说明，他们不对数据丢失负责。事实是，云日志行业必须解决这个问题，才能使其成为较大型消费者的可行选择。

这一领域正以非凡的速度成长和扩张。提供者不断出现，并且都有最新的突破性功能。然而，许多提供者只将重点放在运营上，而不是安全和依从性方面。如果你遇到想要采用的提供者，不要害怕向他们提出有关服务的问题。这是你了解它们所能提供的服务内容的最佳方法。下面是一些需要询问的基本问题：

1）有没有安全主管或者 CSO 能够帮助设置安全态势？

2）采用什么安全控制来保护我的数据？

3）提供者的总体安全态势如何？

4）我的数据和其他人的数据如何隔离？

5）采用什么部署模型？

6）采用何种记账计划？

7）采用什么样的 DR 和业务持续性计划？

8）关于正常运行时间和如下项目的可用性，它们提供何种 SLA：

a）日志获取服务

b）日志留存服务

c）用户界面

d）报告系统

9）下 4 个季度的路线图是怎样的？

10）提供者支持什么样的警报选项（电子邮件、短消息等）？

11）他们是否提供关联？如果有，采用什么形式，统计还是基于规则？

12）和上面的第 8 条类似，他们如何安排维护，换言之，他们是否在每周二凌晨 3 点进行维护？在维护期间，服务的可用性如何？

简单示例：Loggly

我们来简单地看一个例子，它说明如何将日志数据输出到云日志提供者。该示例来自于日志提供者 Loggly（www.loggly.com），它提供了一个基于 Web 的 API 来将数据获取到云中。正如你将要看到的，这是你用于搜索日志数据的机制。第一个例子说明如何使用 Loggly API 的输入类型：

```
# curl -d "this is a test" https://logs.loggly.com/inputs/dccac609-
   d626-4d49-a826-088efc415c77
{"response":"ok"}
#
```

我们使用 curl 命令来简化操作。我们使用的 URL 采用如下形式：

```
httpps://logs.loggly.com/inputs/ [SHA-2 key]
```

每个在 Loggly Web UI 中创建的 API 输入都会生成 SHA-2 密钥。它用于唯一标识这个 API。在例子中，我们发送一个日志消息“this is a test”到云中，得到响应“ok”。这是很好的功能，因为响应容易解析，你可以自动化日志的发送。如果我们想要搜索云，可以用如下命令进行：

```
# curl -u username:password 'http://foo.loggly.com/api/search?q=test'
{
   "data": [
      {
         "timestamp": "2011-08-24T13:55:12.227Z",
         "isjson": false,
         "ip": "206.55.100.193",
         "inputname": "api",
         "inputid": "1735",
         "text": "{\"this is a test\":\"\"}"
      }
   ],
```

```
    "numFound": 1,
    "context": {
        "rows": "100",
        "from": "",
        "until": "NOW",
        "start": 0,
        "query": "test",
        "order": "desc"
    }
}
#
```

这个例子说明了几个要点。首先，搜索和检索用的 URL 和发送数据的不同，它采用如下形式：

```
https://[domain].loggly.com/api/search?q="value to search for"
```

domain 是组织的域名。你在第一次创建账户时设置这个域名。我们的域名是 foo。查询参数是“q”，并遵循标准的 URI 查询字符串格式。我们搜索“test”，返回一个 JSON 响应，显示日志消息“this is a test”。JSON 是默认的返回格式，但是使用格式化的 URL 参数，你可以指定 xml 或者文本返回格式。

最后，如果你的搜索没有返回任何内容，结果如下面所示：

```
# curl -u username:password 'http://foo.loggly.com/api/search?q=foo'
{
    "data": [],
    "numFound": 0,
    "context": {
        "rows": "100",
        "from": "",
        "until": "NOW",
        "start": 0,
        "query": "foo",
        "order": "desc"
    }
}
#
```

云日志提供者应该支持这样的定制 API。为什么这样的 API 有用？下面是一些理由：

1）你可以从自己控制的任何系统或者应用程序发送日志数据，这些系统和应用程序可能不支持 syslog。

2）你可以构建自己的警报系统。通过定期查询云，你也可以编写脚本搜索各种关心的

事件，然后发送电子邮件、短消息等。另一个不错的副作用是，你甚至可以与内部工作单系统（如果有的话）接口。

3）相互制衡。例如，你可以计量发送到提供者的数据量，核对记账的数额。你甚至可以检测发送到云的数据有没有及时到达，在失败时（从提供者那里得到一个错误的响应）重新发送。

本章后面的 21.8 节提供了云日志提供者的一些细节，可以作为进一步研究和探索的出发点。

21.4 监管、依从性和安全问题

到现在为止，你可能觉得云日志非常美妙，它也确实如此。但是很明显，考虑到云日志提供者部署自己的软件和服务，因此需要解决监管、依从性和安全问题。确实如此，许多法律法规的要求都没有面向云计算。因此，作为云消费者，你需要理解，下面这些事情由你负责（云安全联盟，2009）：

- 使用指定云服务的监管适用性。而且，不要忘记跨界问题，例如欧盟的数据放在美国等。
- 云提供者和云消费者之间的依从性职责划分。
- 云提供者生成依从性所需证据的能力。
- 云消费者在弥补云提供者和审计人员 / 评估人员之间鸿沟中的角色。

确保你和内部审计团队（如果有的话）一起工作，协调到云提供者的过渡。

至于 IT 安全，云提供者的安全态势实际上和常规组织没有什么不同。当然，考虑到虚拟硬件的使用、数据隔离方面的考虑和身份管理等，它们需要处理额外的问题。我们来了解一些云提供者应该遵循的最佳实践：

- 内部人员（提供者内部）可能泄露你的数据。
- 云提供者应该在招聘过程中包含背景调查。
- 云提供者应该实施安全策略、控制等，这些措施应该与要求最严格的客户相符。
- 协议或者服务条款应该概述安全策略、备份和恢复、业务持续性等细节。
- 云提供者应该在接收到请求时，提供概述提供者 IT 安全态势的文档。

从依从性和安全的角度，很容易理解使用公共云提供者的含义。在这种情况下，使用混合部署模型变得更加有趣，考虑图 21.2。

混合云跨越私有云（消费者场内，包含两个服务器）和公共云（消费者场外，包含三个服务器）。界线变得模糊，因为在这种部署中，消费者在提供者的站点和自己的站点上都使用云资源。有许多因素需要考虑，如：

1）谁负责加固两个站点之间链接的安全？

2）谁负责场内设施的安全？

3）谁负责场外设施的安全？

4）如果泄露发生在消费者端而非云提供者一端，事故响应和升级如何处理？

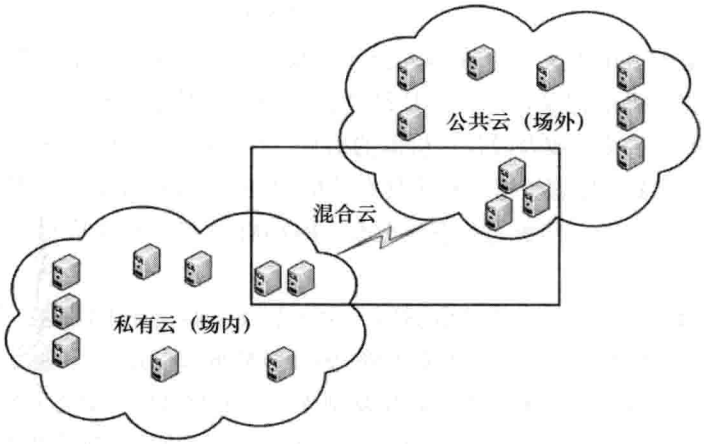


图 21.2 混合部署概图

5）根据组织所处的监管和法规下，你可以参与混合部署吗？

转移到云的任何决策都有许多需要考虑的细节。但是从理解你的责任和需求开始推进云计算，就可以更加清晰地完成过渡。有关云计算的监管、依从性和 IT 安全问题的更详细内容，可以参考云安全联盟（2009）和 Web（2011b）。

21.5 云中的大数据

我们遇到了一个大问题，IT 企业中有许多数据，这一问题很普遍。正如 Stephen Marsland 在他的著作中所提到的，“如果数据有质量，那么地球就是一个黑洞”（Marsland，2009）。这是导致“NoSQL”和大数据运动的原因（参见第 4 章中使用 Amazon EC2 的 Hadoop 示例）。这些运动和因此而兴起的社区，都试图解决如何获取、管理和分析海量数据的问题。

但是，什么是海量数据呢？我们来看一个真实的公司。该公司（为了保护隐私而不列出名称）有超过 5000 台服务器、2000 台路由和交换机，大约 600 台安全设备（防火墙、IDS、IPS、VPN、内容服务交换机、防病毒）和 50 000 个以上的桌面防火墙和主机防病毒软件（桌面和便携电脑）。这至少可以称得上是一个大型网络了。在整个企业中，从所有系统中产生的日志消息每秒大约有 10 万条。我们假设平均日志消息大小为 300 个字节。你可以想想，每小时有多少消息？一天有多少消息？一年有多少消息，我们来做个简单的计算：

10 万条日志消息 / 秒 ×300 字节 / 日志消息	≈28.6MB
×3600 秒	≈100.6GB/ 小时
×24 小时	≈2.35TB/ 天
×365	≈860.5TB/ 年

×3 年 ≈2.52PB

这样的数据量不可谓不大了。许多人会问，收集这么多数据有意义吗？正如本书自始至终所陈述的，我们提倡记录日志，并持续评估所收集的信息。通过这一过程，你不会记录任何不打算查看、不关心或者没什么价值的信息。这通常是很好的原则，但是有些人希望或者必须收集尽可能多的日志数据。对于具有严格的查询和分析需求、且体积持续增长的数据（数据密集型应用程序），传统的关系数据库（RDBMS）已经穷于应付，NoSQL 也就应运而生。

什么是 NoSQL? 这是“下一代的数据库，主要解决一些重要的问题：非关系型、分布、开源和水平可伸缩。它的原意是成为现代化的 Web 级数据库。这一运动始于 2009 年初，正在快速成长。常见的特征有：无模式、简易复制支持、简单 API、最终一致性 /BASE（不是 ACID）、巨大的数据量等。所以，误导性的术语“NoSQL”（该社区现在将其解读为“不仅是 sql”）应该被视为如上定义的一个别名”（NOSQL 数据库，2011）。

上述想法很简单：通过使用高度分布式的架构来实现更高的吞吐率，将数据分为小块（大约 64MB）并将块分配给商用计算机集群。然后，这些计算机处理数据并返回结果。和传统 RDBMS 中扩展机器的技术做个比较，这种技术是在机器上添加更多的 CPU、RAM 和多个输入 / 输出（I/O）通道。例如，具备两个 I/O 通道的机器每秒吞吐量为 100MB，它需要花费 3 个小时才能读取 2TB 的数据。为什么会这样？随着时间的推移，传输率已经有了提高，但是磁盘驱动器的寻道时间并没有同样幅度的提高。因此，许多人（包括云日志提供者）使用实现 NoSQL 概念的系统和软件，提供更快的搜索、索引和其他功能。

简单示例：Hadoop

Hadoop 是一种领先的 MapReduce 实现。MapReduce 技术分为两个步骤，在映射（Map）步骤中，主节点取得输入并将其分解为较小的块。然后，这些小块被分发给工作节点。每个工作节点得到一个答案并将其返回给主节点。在归纳（Reduce）步骤中，主节点取得返回的所有答案并进行某种组合，向客户端返回一个答案。图 21.3 展示了 Hadoop 的逻辑布局。

图中我们有一个商用机器的 Hadoop 云。这些机器代表客户端运行作业（客户端可以是最终用户、应用程序等）。客户端期待的是问题的答案。映射 / 归纳方法效果如此之好的原因是数据和需要在数据上完成的后续分析都可以进行批处理。图 21.3 中的 Hadoop 云是一个分布式系统，允许数据分块且并行处理。Hadoop 和 MapReduce 的另一个核心技术是数据本地化（data locality）。数据本地化处理文件系统上数据的映射步骤。这使得 Hadoop 群集管理员能够配置最接近数据的服务器执行作业。这能够减少网络带宽的消耗，缩短响应时间。

大数据、NoSQL、Hadoop、MapReduce 以及其他相关技术的完整论述超出了本节的范围，请参考《NOSQL Data bases》(2011)。

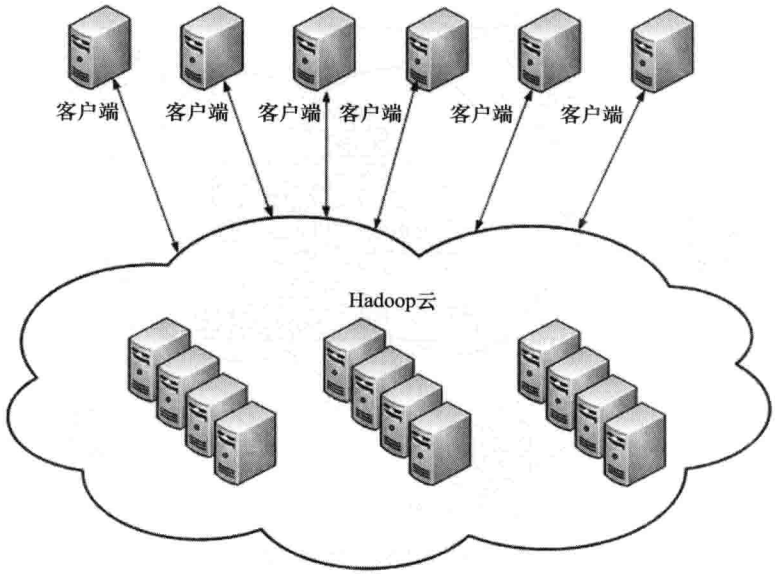


图 21.3 Hadoop 的逻辑布局

21.6 云中的 SIEM

如果不讨论云中的 SIEM，本章就不完整。传统上，SIEM 系统是一个紧缩型套装产品（shrink wrapped）。这意味着你购买软件的许可证，并根据软件的复杂度，向咨询人员付费，由他们花费长时间来安装和配置 SIEM。SIEM 软件公司已经意识到，许多公司不愿意花费数十万美元（或者更多）在 SIEM 部署上。实际上，许多软件公司都在争相翻新软件，使其成为可以在云中部署的产品。

这正是托管安全服务提供商（MSSP）多年以来真正的优势（实际上，在云日志这一术语出现之前，MSSP 就已经在提供云日志了）。MSSP 是承担其他组织网络安全管理任务的公司。他们提供的模型通常有只监控、只管理或者监控与管理。图 21.4 展示了 SIEM 云的逻辑布局。

你首先注意到什么？它看起来和类似我们在 21.3 节中看到的插图。主要区别在于，SIEM 云在功能集上稍微成熟一些。另一个明显的区别是云中的应用程序栈。可以看到，SIEM 云有健全的功能集，它们都可以通过一个或者一组 API 访问。这不仅对于向内部客户及系统（运营、安全运营中心（SOC）、网络运营中心（NOC）、记账、人力资源等）提供服务很关键，对于外部客户同样重要。例如，客户通常希望从提供者向其内部系统输出工作单。大部分提供者实现某种基于 Web 的 API（REST 风格、SOAP 等）。这使得客户很容易编写自定义应用程序代码来获取这些工作单。

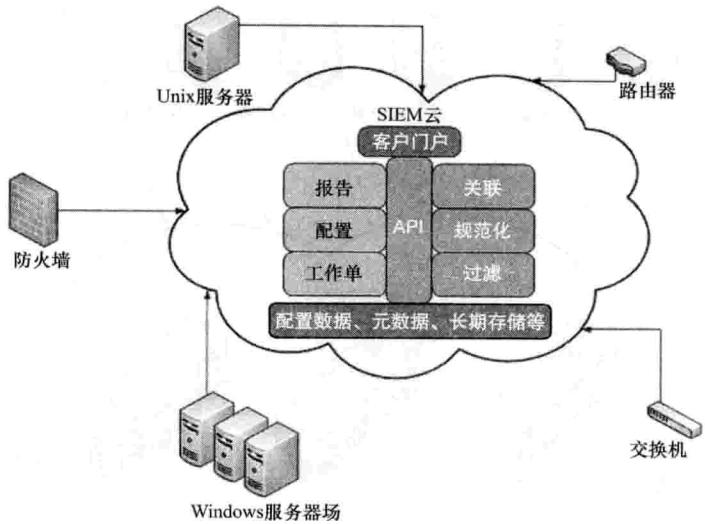


图 21.4　SIEM 云逻辑布局

21.7　云日志的优缺点

现在我们来讨论云日志相关的一些优缺点，这不是一个全面的列表，但是概述了在云日志中常会碰到的优缺点。当然，根据你自身的需求，可能必须进行某些分析，以了解迁移到日志提供者是否有意义。表 21.1 研究了这些优缺点。

表 21.1　云日志的优缺点

优　点	缺　点	对缺点的补救措施
管理：由别人管理硬件、软件 随用随付模型：对你所使用内容的付费 你控制发送的内容和时间		
API 支持：许多日志记录提供者有健全的 API，你可以用它以 syslog 之外的手段向云传输日志数据。你一般还使用这个 API 搜索云数据	API 支持	这可能既是优点，也可能是缺点。如果 API 本身不安全（身份认证和隐私），而且你需要处理监管和依从性问题，这可能是个难点
场外存储：存储的任务由供应商处理。最好的部分是，你一般可以选择保存的时长和数据量	场外存储	是的，这也是一个缺点。根据安全控制、信任矩阵和提供者采用的其他安全相关准则，在云中存储你的日志数据可能不现实。确定提供者是否采用混合部署模型，允许它们整个应用栈的一部分保存在你的数据中心
	安全：安全是一个关注点，因为迁移到云可能产生“优雅的失控”	了解提供者的整体安全态势

21.8 云日志提供者目录

现在我们将为你提供某些日志记录提供者的信息。表 21.2 提供了一些云日志提供者的目录。

表 21.2 云日志供应商目录

供应商	网 站	信 息
Dell SecureWorks	www.secureworks.com	Dell SecureWorks 是一家托管安全服务提供商（MSSP），提供完整的日志留存和管理服务
Loggly	www.loggly.com	Loggy 是全 LaaS 提供商。他们在浏览器中提供搜索控制台，和尝试后购买选项
SplunkStorm	www.splunkstorm.com/	SplunkStorm 是 Splunk 提供的 LaaS 服务，目前处于 beta 测试阶段，但是他们计划提供随用随付记账和其他功能
Sumo Logic	www.sumologic.com	SUMO LOGIC 是一个 LaaS 服务，目标是为消费者提供实时取证和日志管理功能
Log Entries	www.logentries.com	Log Entries 的目标是为大型企业系统的管理、构建和测试提供 IT 问题根源分析工具
Papertrail	www.papertrailapp.com	Papertrail 是服务器、应用程序和云服务的托管日志管理服务

21.9 其他资源

如果你在对云相关信息的搜索中只能访问两个网站，应该访问一下网站：

- NIST 云计算：http://www.nist.gov/itl/cloud/
- 云安全联盟（CSA）：https://cloudsecurityalliance.org。

这是两个具有许多信息和资源的出色网站。当然，互联网搜索可以返回更多的结果。

21.10 小结

本章涵盖了许多素材。需要掌握的要点是：

- 云服务交付模型：三种基本类型是 SaaS、IaaS 和 PaaS
- 云部署模型：四种基本类型是公共、私有、社区和混合。
- 云日志：这是一个年轻的领域，每天都在成长。在你将所有日志数据放到云中之前，一定要了解你和组织的需求。一定要向提供者询问使用中感兴趣的任何问题。
- 云中的 SIEM。MSSP 支持这一模型已经有多年的历史。传统的紧缩型套件 SIEM 供应商开始意识到这一模型的价值，争相在产品中加入云能力和服务。
- 安全考虑：一定要理解提供者实现 IT 安全的方式。还要确保理解在云环境中运营时，你在法律和监管依从性方面所负的职责。

参考文献

Cloud Security Alliance (2009). Security guidance for critical areas of focus in cloud computing V2.1. Web, August 17, 2011. <https://cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf>.

Loggingchallengesandloggingin thecloud–PodCast.LogglyBlog.Loggly,December2010.WebAugust13, 2011. <http://loggly.com/blog/2010/12/logging-challenges-and-logging-in-the-cloud-podcast/>.

Marsland, S. (2009). *Machine learning: An algorithmic perspective*. Boca Raton: Chapman & Hall/CRC. (print)

NOSQL Databases. Web, August 28, 2011. <http://nosql-database.org/>.

Storage as a Service (SaaS). Web, January 02, 2012. <http://searchstorage.techtarget.com/definition/Storage-as-a-Service-SaaS>.

Web (August 20, 2011a). <http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/StandardsRoadmap/NIST_SP_500-291_Jul5A.pdf>.

Web (August 20, 2011b). <http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf>.

10 things to think about with cloud computing and forensics. Web, January 02, 2012. <http://www.cloudave.com/670/10-things-to-think-about-with-cloud-computing-and-forensics/>.

第 22 章 Chapter 22

日志标准和未来的趋势

本章要点

- 从今天推知未来
- 日志的未来和标准
- 展望未来

22.1 概述

20 世纪著名的物理学家 Richard Feyman 曾经凭借简单地预测未来和当下一样，而在许多次打赌中获胜。应该承认，在我们所处的不断变化的 IT 环境中，那种方法似乎是必输无疑的。但是，许多东西都保持不变，在日志记录和日志分析领域中，这一点比其他领域更明显。

IT 管理员和组织在 20 世纪 80 年代末至 90 年代初（那已经是 20 多年前）需要处理的问题，现在仍然保持着近似的形式。顺便说一句，对于许多信息安全问题也是如此，有时候，信息安全问题永远得不到真正的解决。例如，密码错误在将近 50 年以前就被视为一个问题（实际上，这个问题甚至可以上溯到计算机时代以前），但是许多公司今天对同样的问题仍然没有明确的解决方法。本书的作者们从事这一行业多年，日志仍然常被叫做“没有得到利用的财物”，可能从 20 世纪 90 年代至今都是如此，未来是否仍然如此，完全取决于 IT 社区的努力。

组织收集不同 IT 记录（包括日志）的能力在过去一二十年内无疑得到了改善。但是，尽管有服务提供商、昂贵的工具和大大扩张的相关知识库，许多公司仍然在这一领域苦苦挣

扎。日志已经从 IT 领域转移到依从性领域，现在它还要再次转移，达到情报的高度。

我们将从未来开始讨论，并简单地讨论如何从今天推知未来。注意，本章讲述的不仅是日志的未来，也是关注日志并借此采取行动和做出决定的人们的未来。

22.2 从今天推知未来

我们现在简单地概述一下历史，syslog 诞生于 20 世纪 80 年代初，是 Eric Allman 的 sendmail 计划的一部分。应该承认，日志记录和日志分析的历史开始得更早，但是 syslog 的诞生标志着跨越 30 年的革新（1982 至 2012）。

在这一过程中，日志数据的容量显著地增加了。此外，不仅数量增加，日志数据的质量也有了改进。新的日志源，甚至概念性的新 IT 环境（如移动计算、云计算和虚拟化）出现并成为主流。实际上，即使是 PC 也诞生在同一时代，日志记录先于这一重要的进步，现在似乎已经平淡无味，甚至有些陈腐了。

令日志数据产生意义，以日志文件的形式交流高质量的有用信息的是分析方法，而这些方法似乎还没有足够的改进。人们现在不再从 UNIX 系统上查看 syslog，从 Windows 上查看事件日志（20 世纪 90 年代的发明），而是查看更广泛的日志集，包括难以理解的文本和高度结构化的 XML（遗憾的是，这种方法要表达 50 个字节的 syslog 消息，往往需要传输几兆字节的数据）。

22.2.1 更多的日志数据

我们的第一个推断就是会有更多的日志数据。这分为几个方面，一个比一个危险。

无疑，在以后的 5 至 7 年甚至更长的时间内，日志数据的容量将会增大。越来越多的计算资源联网，其中许多都会产生日志。监管的压力驱使组织增加系统和应用程序日志记录的容量。

除了简单的容量增加之外，在产生日志的系统数量上也会有增加。处理一两个系统总是很容易的，而处理几千个系统就相当有挑战性。不过可以预料，在即将出现的“物联网”上，许多不同系统联网，加上移动和云计算的增长，将造成容量和日志记录实例数量同时激增。

除了生成日志的系统数量的增加，现有系统预计也将生成更多的日志数据，或者生成更详细（因而更大）的日志数据。例如，几年之前记录失败登录访问企图的系统，将会记录成功的企图和其他活动。如果同样的系统在一两年内记录对敏感文件的所有访问，作者也不会感到惊讶。我们还预计，在几年内，更多的活动将会建立审计跟踪，需要进行分析、留存并对某些情况采取行动。

还有一个趋势，即通过日志进行更多的访问和活动的审计，例如，文件服务器、存储或者数据库供应商以前不太关心日志，但是现在他们关心这一点（有些已经关心，有些开始这么做了）。以前人们只关心对数据的访问，而现在发展成为可审计的数据访问。

监管和调查的压力导致组织保存日志更长的时间，从而增加了留存数据的容量。即使日志源的数据相同，产生的日志数据容量相同（这一假设几乎是不可能的）由于留存期的加长，也会导致存储和收集的日志容量增大。几年以前，组织保留超过 3 个月日志的情况并不多见，现在，保存一年的情况显著增多。在许多情况下，更长的留存期不是由法规和外部强制造成的，而是调查组织内部威胁、内部人员活动、政府发起的威胁和其他高级恶意活动的需要。

数据容量的增大导致分析和理解的困难。这对于某些读者来说可能不明显，但是许多任务在日志数据为 10MB 时微不足道，在 1GB 的日志数据时也很简单，但是当数据达到 PB 级别，就变成了极大的挑战。必须承认的是，今天的组织中必须分析 1PB 日志数据的组织不多，但是这毕竟存在，而且数量正在增长。在这种容量水平下，即使简单地复制数据也会成为工程和架构上的挑战，更不要说让数据通过分析引擎了。

与此同时，应用程序日志的容量也有大规模的增加，它们和基础设施日志使用同一个系统分析，和基础设施日志一样，应用程序日志也将发生转变。对于某些云日志场景来说（在第 21 章中讨论过），应用程序日志记录是组织所拥有的唯一日志记录。从企业传统应用程序、云应用程序到移动应用程序，所产生的日志数据预计都会有增加。

而且，IT 日志记录可能和 OT（运营技术）以及 CT（通信技术）的日志记录相伴。从移动电话提供商设备到电梯和大楼控制系统，日志数据的世界将会扩张，超出 IT 的范畴，从而相应地增加数据的容量和多样性。

今天，没有多少人知道移动应用程序也会产生活动的审计跟踪。随着企业对各种移动技术使用的扩展，分析这类设备日志的需求在将来的几年内就会出现。这些日志首先将用于运营目的，还将可能用于安全目的并纳入各种法规的规定。

总是有些人断定“大数据”的时代已经来临。实际上，许多组织还在“小数据”中挣扎。如果你的组织拥有的工具无法分析 10GB 数据，也就没有必要在辞典中加入“PB”这个词。现有的工具只能帮助人们了解大数据的讨论发生之前环境中的情况。首先处理小数据，建立你自己的知识库，然后再考虑大数据的问题。许多组织被一行日志、一个记录搞得不知所措，对处理几万亿条这样的消息自然没有任何准备。

22.2.2　更多动力

我们的第二个推断是认真对待日志有更多的动力。分析人员估计，大部分日志分析和管理工具（包括商业化的 SIEM 工具）被部署用于监管依从性。但是，关于日志管理方法的说法很模糊，这可能导致混乱。尽管如此，依从性在近年以来大大推动了日志管理和日志分析的进步，预计还将继续成为改进的动力。我们在其他章节中讨论日志记录和依从性。

我们预测，更多的强制管理、控制框架、法规和其他外部指南将迫使组织扩展日志数据的处理。从这些数据中获得价值需要时间和精力——这在内部 IT 组织中往往很缺乏。首先，这些工作看起来很乏味，这是因为数据容量很大且具有主观的特性。尽管有这些挑战，日志记录是 IT 可审核性的主要手段，因为大部分用户和系统操作可能记录在日志中。这也就是

日志记录成为完善的依从性技术，纳入许多法规和法律规定的原因，在可预见的未来中，这仍将持续。

认真考虑日志的其他动力来自于组织控制力较弱的环境，在这些环境中，组织需要用更多的可见性来弥补控制力的不足。我正在盯着你呢，云！可以预见，监控部署在直接控制之外的资产需要更多的监控和可见性功能，这些往往依赖于日志。

22.2.3 更多分析

我们的第三个推断是有争议的（尽管不应该如此），即更多的分析方法，和更多获得日志数据价值的方法。许多人预测，我们现在正处于许多数据分析技术的关键临界点上，包括为安全所做的日志数据分析。有些行业观察家认为，过去，一个简单的 Netflix 影片选择操作在后端触发的算法智能，就超过所有日志分析产品的总和。在许多方面，我们正在达到现有分析方法的极限。

可以预期，新的分析方法将会出现，而现有的方法，从基于规则的关联到统计分析，再到数据挖掘，将更加广泛地用于分析日志数据。这些算法必须满足诸如分布式日志分析（如源于云应用程序的组件）、任何规模、任何用户、任何系统、任何日志数据的分析等需求，也可以用于多种目的。

除此之外，其他类型数据（如上下文数据）在未来可能更广泛地和日志数据一起分析。需要有从小型到企业级全范围内的工具，帮助人们理解各类数据。

这一预测称不上是个推断，但是所关注的也是新的分析时代带来的新功能和潜力。

除了上述推断之外，从当今信息技术的更新浪潮中，也一定会产生新的挑战。人们可以将部署在公共云提供者环境中的应用程序视为远程应用程序，但是实际上云计算带来了一些新的技术，也带来了影响日志产生和利用的新运营实践。

总的来说，最近媒体关于“大数据”(尤其是“大数据”分析，而不只是 PB 级数据的处理）的鼓噪与日志数据直接相关。许多公司已经开始使用 Hadoop 和其他大数据相关技术收集和分析日志数据。在某些情况下，它是在许多客户的云中完成的，而其他一些则是在场内完成的。

22.3 日志的未来和标准

挑战越来越多，那么日志记录和分析的未来能够带来什么好处呢？主要的因素是日志的标准化。通用事件表达式（Common Event Expression，CEE）[⊖] 等日志标准有可能显著地改变未来的面貌。

但是，为什么是现在？在没有 IT 日志记录标准化的情况下，我们已经度过了 30 多年。日志记录的老将 syslog 将会何去何从？

⊖ http://cee.mitre.org/

RFC3164 中引为“标准”的 syslog，实际上是一个“不是标准的标准、”尽管 RFC 文档中充斥着“从一开始，人类的生活依赖于消息的传输。对于有自我意识的有机单元，这些消息能够传达许多不同的含义”之类的文字，提醒我们 syslog 创建于加州伯克利，那可能是美国的“大麻之都”。接着“由于每个进程、应用程序和操作系统都是独立编写的，syslog 消息的内容并不统一。因此，对于消息的格式和内容没有任何推断。”最后这一句话明确地说明，它并不是一个标准，只是用 UDP 协议将日志从 A 位置移到 B 位置的一种方法。Marcus Ranum 曾经把 syslog 称作“人们所知的最糟糕，也是最常用的日志记录架构。”[⊖]

今天，通用事件表达式（CEE）已经形成草案规范。这一标准有可能终结模糊、有歧义和无结构的日志数据。相反，它将用人类可读和机器可读、高效、简洁而没有歧义的日志来代替。这一标准草案试图影响日志记录生态环境的所有方面（见图 22.1）。

确切地说，它将定义日志如何生成、传输、分析等。在本书编著的时候，发布了如下规范：

■ CEE Architecture Overview Specification，版本 1.0 α

CEE 概要提供了 CEE 的高层概要，以及总体架构的细节，介绍了各个 CEE 组件，包括 CEE Profile、CEE Log Syntax 和 CEE Log Transport。CEE 概要是一组文档和规范中的第一个，这组规范组合起来，提供创建完整的 CEE 事件日志标准的必要部分：

■ CEE Profiles Specification，版本 1.0 α

CEE 配置允许用户定义事件结构和提供的数据，改进事件数据的解读和分析。配置由三个组件组成，分别为 CEE 兼容事件提供标准化字段字典、事件分类和基本需求：

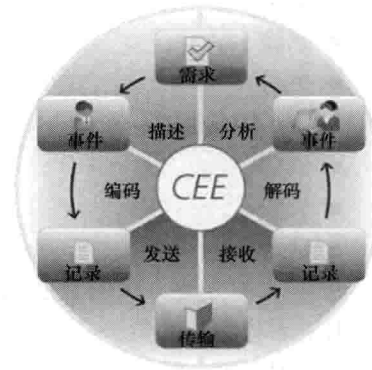


图 22.1　CEE 影响整个生态系统
http://cee.mitre.org/docs/overview.html

■ CEE Log Syntax（CLS）Specification，版本 1.0 α

CLS 规范描述通用事件表达式（CEE）事件的编码和解码需求，并为 XML 和 JSON 事件记录提供编码声明：

■ CEE Log Transport（CLT）Specification，版本 1.0 α

CEE 日志传输（CLT）提供安全可靠的日志基础设施所需的技术支持。CLT 规范定义兼容的日志传输必须满足的需求列表，此外，CLT 定义传输映射，定义通过某种协议传输编码的 CEE 事件记录的标准方法学。[⊜]

CEE 还定义了统一日志消息含义的分类，允许开发人员无歧义地定义特定事件所揭示的情况，以及如何在日志中记录，这种分类将包含在事件配置规范中。分类的关键思路是用字段定义每个事件：对象（事件发生在哪里？），操作（发生了什么？）和状态（结果是什么？）

⊖ http://www.scribd.com/doc/94205026/16/Syslog-RFC-3164./

⊜ http://cee.mitre.org/about/documents.html

(O-A-S)。分类的早期版本可以在 http://cee.mitre.org/repository/schemas/v10alpha/taxonomy.xsd 上找到。

对象取值包括：

- 系统
- 文件
- 用户
- 目录
- 连接

操作取值包括：

- 打开
- 关闭
- 删除
- 更新

最后，状态包括：

- 成功
- 失败
- 延迟

思考这种分类方法的最实用方法是将每个日志看作完整的句子：发生了什么，发生在哪个对象上，结果如何？

因此，标准可以容纳所有必要的领域标准化。参见图 22.2。需要记录什么日志、如何记录日志、记录哪些细节、如何将日志从生成系统传输到分析系统。

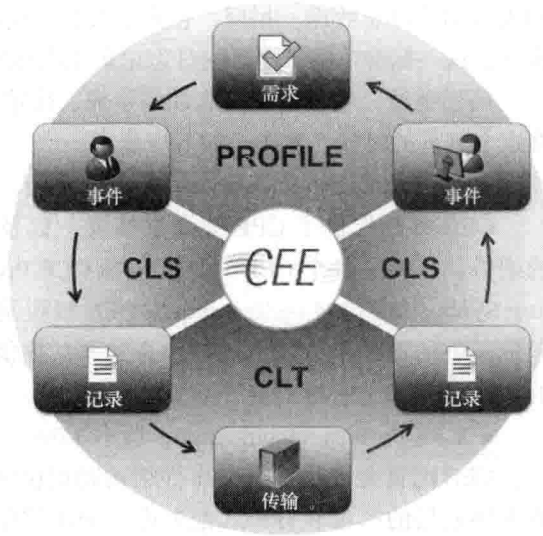


图 22.2　领域标准化，
http://cee.mitre.org/docs/overview.html

即使在今天，开发人员也可以选择使用 CEE 格式，并创建如下的日志：

我们来看一个 CEE XML 示例：[⊖]

```
<?xml version="1.0" encoding="UTF-8"?>
<CEE>
   <Event>
      <crit>123</crit>
      <id>abc</id>
      <p_app>application</p_app>
      <p_proc>auth</p_proc>
      <p_proc_id>123</p_proc_id>
```

⊖ http://cee.mitre.org/docs/cls.html

```
        <p_sys>system.example.com</p_sys>
        <pri>10</pri>
        <time>2011-12-20T12:38:05.123456-05:00</time>
        <Type>
            <action>login</action>
            <domain>app</domain>
            <object>account</object>
            <service>web</service>
            <status>success</status>
        </Type>
    </Event>
</CEE>
```

接下来我们看一个 CEE JSON 示例，它可以用于 syslog 传输。

```
{"Event":{"crit":123,"id":"abc","p_app":"application","p_
proc":"auth","p_proc_id":123,"p_sys":"system.example.
com","pri":10,"time":"2011-12-20T12:38:05.123456-05:00",
"Type":{"action":"login","domain":"app","object":"account",
"service":"web","status":"success"}},"Module":[{"Augment":{
"p_proc":"my_process","p_proc_id":123,"p_sys":"syslog-relay.
example.com","time":"2011-12-20T17:41:12Z"}},{"Augment":{"p_
proc":"my_process","p_proc_id":123,"p_sys":"syslog-relay.example.
com","time":"2011-12-20T17:41:12Z"}}]}
```

这些日志可以很容易地由任何当前的日志分析系统消费，轻松地与商业化的 SIEM 工具集成，如果使用 JSON，分析人员理解起来也很简单。特别是，JSON/ 文本日志和创建其他的“名称 = 取值”配对格式一样简单，比传统的自由格式日志更实用。

这些日志可以存储在关系数据库，或者现代化的基于 Hadoop 或者其他 NoSQL 存储设施中。

标准采用趋势

考虑到过去标准化日志所经历的惨痛失败，CEE 的采用也无法确定。怎样才能让人们接受这一标准？日志生成者、消费者、政府和监管部门能做什么？

从战略上说，标准的采用可能发生在多个机制上：

- MITRE 与美国政府采购合同关系紧密，这些合同可以用来推进标准的采用，就像几年前的公共漏洞与暴露（CVE）漏洞格式那样。
- 较小的软件供应商可以用 CEE 代替自己的新格式，在 CEE 完成时实现标准依从性。
- 开源日志记录工具有利于从原生的 UNIX 日志和其他 syslog 转换为结构化的 CEE（JSON 格式）。

现在，rsyslog（在第 15 章中介绍）和 syslog-ng（在第 5 章中介绍）等开源日志记录工具构建了自己的 CEE 支持。多个商业化 SIEM 供应商也在其工具内部使用 CEE 格式和思路，

使它们在某些日志以此为原生格式的环境中更容易应用。操作系统供应商也正在将注意力放在它们的格式与 CEE 的转换器上。

22.4 渴望的未来

除了预测以外，我们还将尝试定义人们所渴望的未来将是什么样子。下面是在未来几年内日志记录的世界中我们乐见的一些现象：

- 日志标准得到日志生成者的广泛采纳。应用程序和设备制造商以及云平台制造商都以标准格式生成日志。日志数据的歧义和将日志看作断续的英语句子的需求一去不复返——希望永远如此。
- 应用程序开发人员在编写应用程序时立即采用日志标准和标准库及 API，实现实用的日志。形成的数据不仅基于标准，而且全面、不过分臃肿，可用于大部分运营、安全和监管需求。
- 总而言之，我们将拥有更加透明、直观和可审核的系统，不管采用的是移动、云、虚拟还是传统方式。正如 Dan Geer 曾经说过的："问题是，这些技术（访问控制，AC）不能扩展，如果你试图对每秒出现的大量新数据进行更精细的控制，就会带来祸及安全的复杂性。真正扩展的是可审核性。在一个自由的国家里，你不需要为大部分事情要求授权，但是这种自由是由某些知识支撑的，如果你有更多的要求，就必须付出代价。"[⊖]
- 协助分析日志的工具必须更加智能，同时更加易用。这些工具可以消费数据，不管数据是遵循标准的还是遗留的，并且根据数据生成有用的建议和结论，而无须太多的人工调整和交互。但是，希望与工具交互的人绝对可以这样做，甚至因此得到更好的结果。
- 更多的公司提升自己的日志成熟度曲线，更多组织在分析和理解日志数据上变得更有能力。这种能力并不廉价，但是上面所提到的工具和日志的改进，会使它们变得更容易承受。
- 最后，在 IT 领域内外的人在工具、改进的日志和改进的成熟实践和过程的支持下，将日志用于更多的目的，这些支持手段专注于获得更多对业务和技术以及系统保护、法规依从性有益的见解。

22.5 小结

组织如何为未来做准备？用一位著名科学小说家提出的原则：未来就在那里，只是它分

⊖ http://geer.tinho.net/geer.housetestimony.070423.txt

布得不均匀。

例如，你可以选择等待，直到日志标准成为主流，受到每个软件制造商的支持（几十年之后）。我们可以使用不断发展的标准中的素材，创建自己的组织标准，从未来的全球标准得到灵感。这将使你获得一些价值，生成更实用的日志，告诉你更多有关所发生情况以及系统、网络和云的信息。

你不必等待更加智能的工具出现和普及，可以编写在某些你所需要的方面上更加智能的工具。

不用担心大容量的日志，开始分析你所拥有的数据，以便获得在小规模日志数据（现在这意味着几个 GB 的数据）上的专业能力，为未来 10 倍到 1000 倍的容量做好准备。

从根本上说，对于未来需要的是准备，而不是恐惧。未来会给你和组织的日志记录和日志分析带来有益的进步。

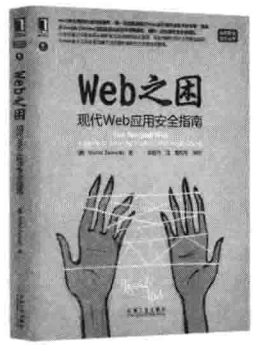
推荐阅读

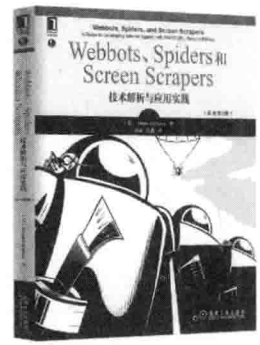

安全软件开发之道
构筑软件安全的本质方法
Building Secure Software

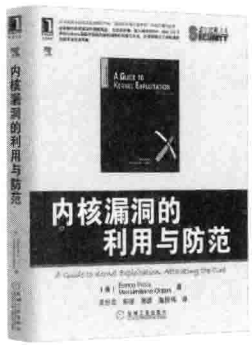

数据隐藏技术揭秘
破解多媒体、操作系统、移动设备和网络协议中的隐秘数据
Data Hiding

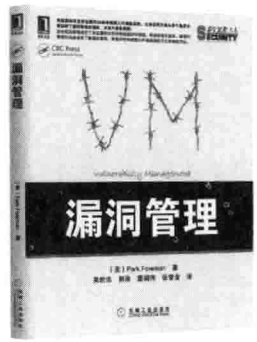

Rootkit
系统灰色地带的潜伏者
（原书第2版）
The Rootkit Arsenal

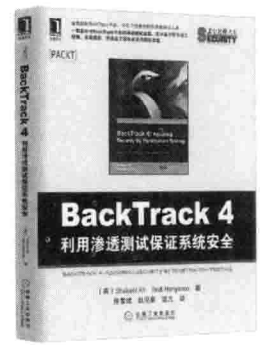

恶意软件、Rootkit和僵尸网络
Malware, Rootkits & Botnets
A Beginner's Guide


Web之困
现代Web应用安全指南


Webbots、Spiders和Screen Scrapers
技术解析与应用实践


内核漏洞的利用与防范


漏洞管理


BackTrack 4
利用渗透测试保证系统安全

[General Information]
书名=日志管理与分析权威指南
作者=（美）褚瓦金，（美）施密特，（美）菲利普斯著
丛书名=华章程序员书库
页数=３１６
ＳＳ号=１３５８９０４２
出版日期=２０１４．０６
出版社=北京：机械工业出版社
ＩＳＢＮ号=９７８-７-１１１-４６９１８-６
中图法分类号=ＴＰ３９３．０７
原书定价=６９．００
参考文献格式=（美）褚瓦金，（美）施密特，（美）菲利普斯著．日志管理与分析权威指南．北京：机械工业出版社，２０１４．０６．
内容提要=本书从日志的基本概念开始，循序渐进讲解整个日志生命期的详细过程，涵盖日志数据收集、存储分析和法规依从性等主题，并通过丰富的实例，系统阐释日志管理与日志数据分析的实用技术和工具。