第 24 章 探究恶意软件沙盒

基于签名的恶意软件检测在应对混淆、加壳与加密的恶意软件时变得无力，尤其是在静态分析的场景下，攻击者的对抗行为会让大多数基于签名的检测失灵。不仅如此，随着恶意软件日渐变得更加复杂，不仅检测恶意软件变得困难，分析和调试恶意软件也变得更加困难。为了应对这些检测中面临的困境，包括反病毒引擎在内的反恶意软件解决方案还会检查系统上进程的行为，寻找任何可能表明系统上存在恶意软件感染或者恶意攻击活动的痕迹。

沙盒作为一种基于行为的动态分析技术，也就应运而生。它不仅被恶意软件分析人员广泛使用，也几乎被嵌入在当今几乎所有反恶意软件解决方案中。本章将要介绍为什么使用恶意软件沙盒，以及沙盒的各种组件。

什么是恶意软件沙盒？

恶意软件沙盒是一个受控的、隔离的分析环境，利用其执行样本文件以记录样本在执行过程中的所有行为。被记录下的事件与行为会被上报给沙盒的使用者，分析人员可以据此对恶意行为进行分析。在大多数情况下，沙盒需要使用虚拟机来实现。当然，也可以使用物理机来创建沙盒系统，这被称为硬件恶意软件沙盒。

与此前分析恶意软件时使用的 APIMiner 类似，恶意软件沙盒主要用于从样本文件的执行行为中提取 API 执行日志。除了 API 执行日志外，沙盒还可以从其他角度来观察与记录样本文件的行为。例如，沙盒可以借助 ETW 等事件跟踪工具或使用内核驱动程序来监控恶意软件的系统级行为，使用这种方法也可以监控恶意软件嵌入的内核态组件。以下是沙盒监控与记录的 API 日志与行为事件类型：

- 进程与线程

- 注册表
- 文件与目录
- 网络
- 服务
- Synchronization
- 系统
- 界面

一旦将沙盒记录的 API 日志与事件上报给沙盒的使用者，就可以利用这些数据进行签名检测、启发式检测与其他检测算法，对样本文件的恶意性进行分析与判断。整个过程如图 24- 1 所示:

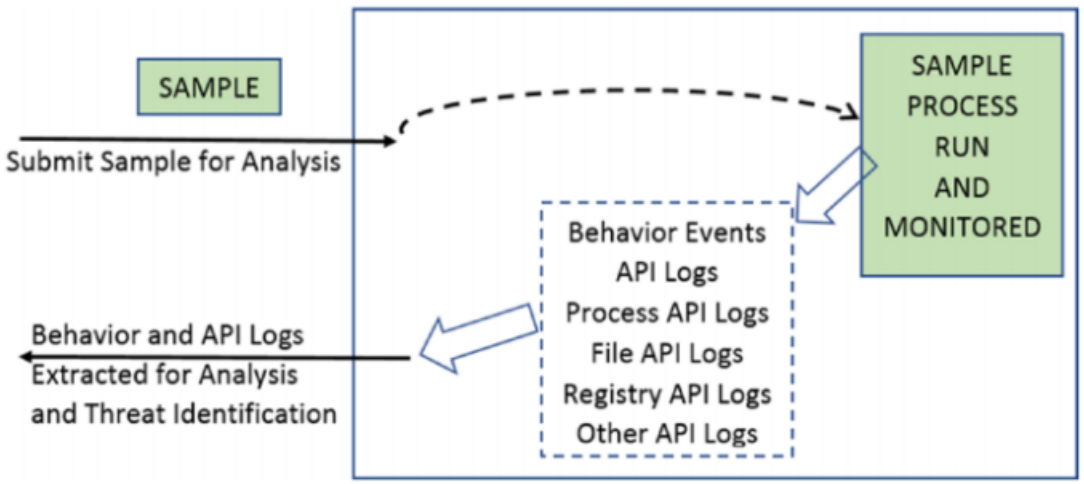

图 24- 1

正如之前所述，恶意软件沙盒通常使用虚拟机来实现，虚拟机通常位于运行虚拟机的管理程序（hypervisor）或者模拟器（emulator）的主机操作系统上。管理程序或模拟器上可以运行单个沙盒（虚拟机），也可以运行多个沙盒（虚拟机）。大多数商业沙盒产品都会部署多个管理程序来运行沙盒（虚拟机），以整个集群的形式对外提供服务。分析集群式的沙盒能够支撑反恶意软件解决方案在高负载场景下并行处理恶意样本，如图 24- 2 所示。

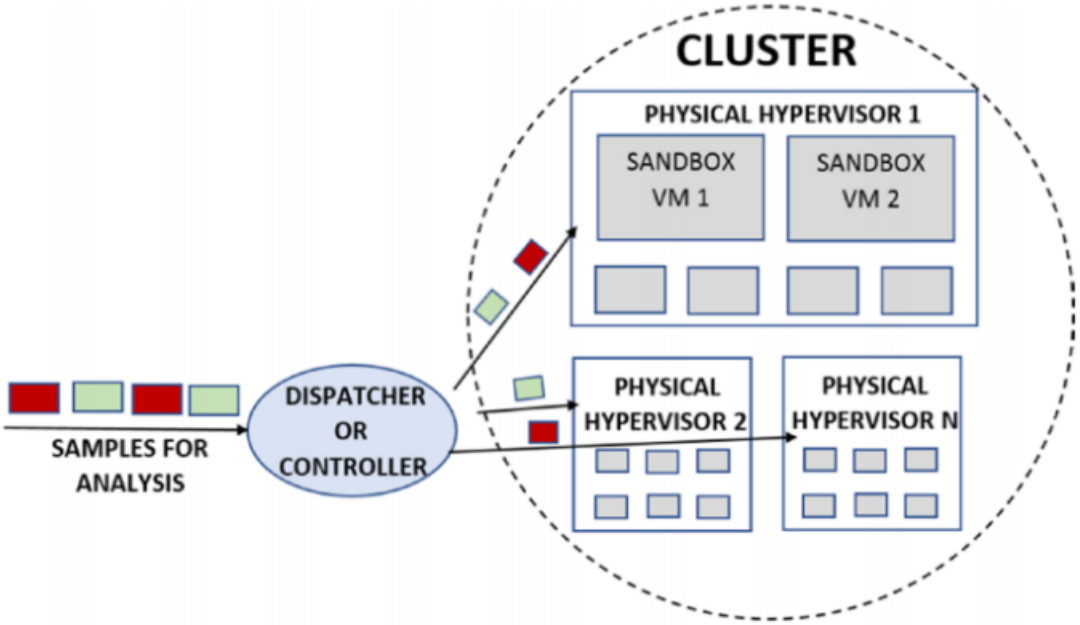

图 24- 2

为什么要使用恶意软件沙盒?

每个用户使用恶意软件沙盒的原因各有不同，下面列出了部分常见的原因:

- 达成目标并提高检测效率。基于动态行为的威胁检测是判断样本文件是否为恶意软件的重要组成部分，几乎所有反恶意软件解决方案都会使用沙盒。沙盒不仅能够帮助提高反恶意软件产品的检测效率，也能为分析人员的分析与调试工作提供帮助。
- 部署受控且安全的分析环境。沙盒为用户提供了一个受控的、隔离的系统，使用该系统可以在不必担心恶意感染的情况下监控样本文件的动态行为。
- 构建高速、高效的自动化分析环境。现如今几乎所有反恶意软件解决方案都在使用沙盒，将样本文件下发到沙盒中进行分析，分析完成后回收分析结果，这一切都可以自动完成。在沙盒的帮助下，自动化的恶意软件分析成为可能。基于沙盒可以快速分析样本文件，从而提高检测效率。
- 便于分析人员调试恶意软件。不仅反恶意软件解决方案在使用沙盒，分析人员也会使用沙盒来分析样本文件。此前使用 APIMiner 来记录恶意软件的 API 行为，这些数据

可用于样本分析。类似 APIMiner 的同类工具，都与恶意软件沙盒核心组件—API 记录模块相仿。使用 Cuckoo 等恶意软件沙盒，将样本提交沙盒进行分析就可以获取 API 行为日志。

安全架构中沙盒的定位

沙盒是各种恶意软件检测解决方案中非常重要的组成部分。厂商不仅可以利用本地资源运行沙盒，还可以使用云资源部署沙盒。这可以为用户提供全球可达的沙盒解决方案，也可以将威胁防御产品推广到世界各地。

在自研威胁检测与防护系统时，如果系统涉及到文件（几乎肯定会与文件相关），就可以考虑在架构设计中应用沙盒。以下是将沙盒集成在产品设计中的一些典型案例，如图 24- 3 所示。

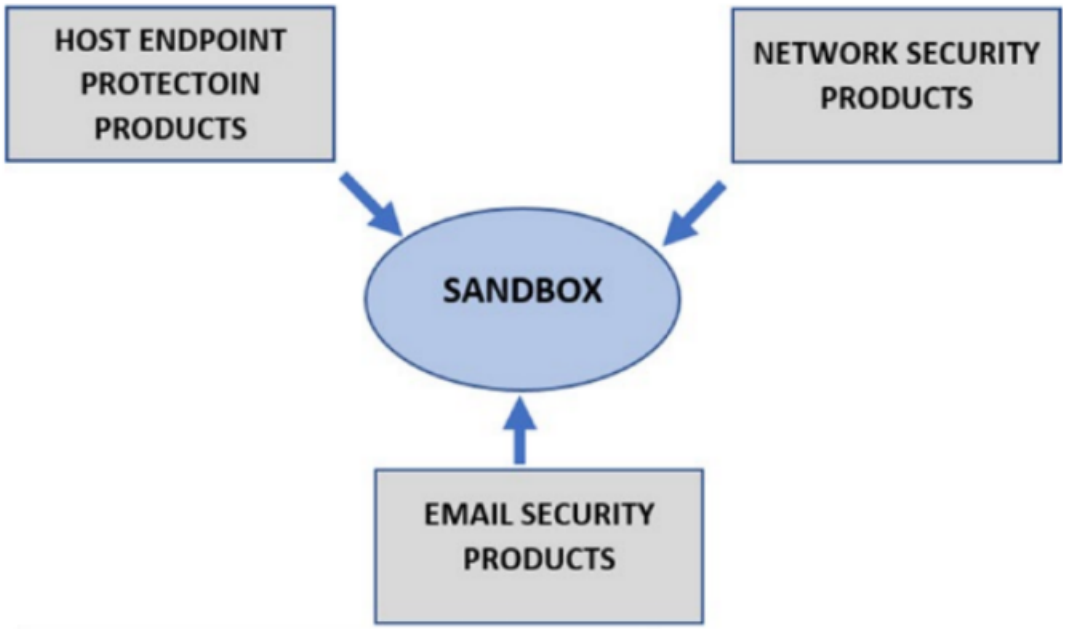

图 24- 3

- 网络安全产品。在第 23 章中介绍过，文件提取是 IDS/IPS 的重要功能。众所周知，提供防火墙/IDP/IPS 等网络安全产品的厂商会利用文件提取功能来获取网络中传输的文件，并将这些文件提交到沙盒进行分析。

- 端点安全产品。端点代理、端点数据记录工具甚至是一些端点防护产品，都会利用沙盒来分析在端点上获取的样本文件。
- 电子邮件安全产品。很多电子邮件通常会包含附件，其中一些攻击者发送的恶意电子邮件就包含恶意附件。电子邮件安全产品会不断监控往来的电子邮件，从中提取附件并进行分析，分析手段就包括将文件提交到沙盒中。

沙盒设计

以下是沙盒的一些典型组件:

- 沙盒虚拟机
- 主机控制模块
- 沙盒控制模块
- 监控模块
  - API 记录模块
  - 内存转储模块
- 欺骗模块
- 主机与沙盒之间的通信信道

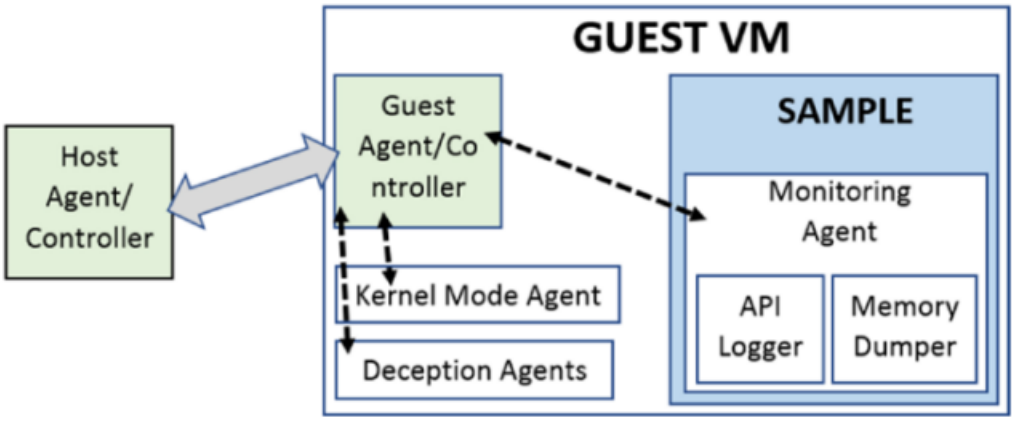

图 24- 4

图 24- 4 显示了沙盒的主要组件以及各个组件间是如何进行交互的。在下一节中将会介绍沙盒的工作流程，从提交样本到获取行为日志的分析全过程。

样本文件分析工作流程

沙盒的任务是分析样本文件并观察其行为，最后将分析结果上报给使用者，使用者可以根据相关的事件和日志进行分析。接下来将要介绍整个沙盒的分析工作流程，读者可根据需要回看图 24- 4。

1. 样本文件提交给主机控制模块。
2. 主机控制模块通过静态分析确定沙盒所需的操作系统和分析环境设置。例如，如果样本文件为 PE 可执行文件，就需要在 Windows 操作系统的沙盒中运行。如果样本文件为 ELF 可执行文件，就需要在 Linux 操作系统的沙盒中运行。稍后会进行详细讨论。
3. 根据样本文件的执行需要，主机控制模块从集群中寻找满足条件的、空闲的沙盒虚拟机。如果所有沙盒虚拟机都在运行，就会等待资源可用。
4. 主机控制模块获得可用资源后，会将沙盒虚拟机恢复到原始状态的快照并恢复虚拟机的执行。
5. 沙盒虚拟机启动后，主机控制模块要与在沙盒虚拟机中运行的沙盒控制模块建立通信信道。
6. 建立通信信道后，主机控制模块将（1）中的样本文件传输给沙盒控制模块，后者运行样本文件并返回行为日志的分析结果。
7. 沙盒控制模块运行样本文件并注入监控模块，后者通常是一个 DLL 文件。监控模块包含多个功能的组件，例如 API 记录模块与内存转储模块。利用这些组件，沙盒可以记录样本文件使用的各种 API，或者从正在运行的进程中转储内存。

8. 沙盒控制模块从（7）中获取的 API 日志和内存转储等相关数据，通过（5）中建立的通信信道或者通过其他方式回传给主机控制模块。
9. 沙盒控制模块将行为日志与数据返回给沙盒使用者。

这样也就完成了整个分析过程！接下来的几节将会详细分析各种模块的内部结构，并详细介绍实现一个成熟的沙盒所需要的条件。

沙盒虚拟机

沙盒虚拟机是沙盒的核心组件，执行样本文件并监控执行行为都在其中完成。沙盒虚拟机通常会在部署沙盒的时候进行完整配置，并留存快照以供进行动态分析。

虚拟机配置

为了支持分析不同的样本文件类型与格式，沙盒通常会部署多个虚拟机，每个虚拟机采用不同的配置来满足不同的需求。根据样本运行的目标操作系统，需要配置 Windows、macOS 与 Linux 的虚拟机。

根据想要分析的文件类型，还需要在每个虚拟机中安装各种对应的软件。例如，如果沙盒需要分析 Microsoft Office 文档文件，就需要在沙盒中安装 Microsoft Office 等可以运行这些文件的相关软件。如果沙盒需要分析 PDF 文件，就需要在沙盒中安装 Adobe PDF Reader、Foxit Reader 等软件。诸如此类，要分析的文件类型及上下文决定了沙盒虚拟机的操作系统以及环境中需要安装的工具。

模拟实际用户

在第 2 章中，已经介绍了如何设置分析虚拟机来模拟常规的实际用户。同样的，也应该用相似的方式来配置沙盒虚拟机模拟实际用户。

几乎所有攻击者都清楚，大多数安全厂商都会使用沙盒来分析恶意软件的动态行为。为了阻碍安全厂商通过沙盒进行动态分析，攻击者会在编写恶意软件时应用各种 armoring（详见第 19 章）与反分析技术。这些对抗技术旨在检测样本文件是否在分析虚拟机或沙盒虚拟机中运行，确定在分析环境中后会表现出良性行为或者提前终止运行。这样会导致分析日志中不存在任何能够表明恶意目的的行为，从而误导分析人员与检测引擎。

为了应对各种对抗技术并防止恶意软件发现其在分析环境中运行，需要使沙盒虚拟机的操作系统与环境模仿实际用户的操作系统、硬件条件和相关环境。如果能够欺骗恶意软件，让其认为正在失陷主机中运行，恶意软件就会展现其攻击意图。分析人员也就可以通过沙盒虚拟机来提取恶意行为日志，从而辅助检测样本文件是否为恶意。

除了在第 2 章中“模拟实际用户”章节中介绍的内容（请确保返回阅读）外，以下也是在构建沙盒虚拟机时要考虑的要点。

- 键盘与鼠标移动。实际用户通常会使用键盘和鼠标在系统上进行各种操作，但由于沙盒虚拟机是一种自动化分析系统（交互式恶意软件沙盒除外，相关内容读者可以自行查阅），不会有用户来操作键盘与鼠标。恶意软件会将键盘与鼠标移动的缺失，作为识别沙盒的一个特征。为了解决该问题，许多沙盒会在环境中模拟光标移动与键盘敲击，以欺骗恶意软件使其认为实际用户正在使用该系统。
- 隐藏分析工具。大多数实际用户都不会安装任何恶意软件分析工具与框架，这些工具通常安装在分析环境中。众所周知，恶意软件也会通过检索此类工具的存在，来判断是否是否在分析环境中运行。分析环境中的分析工具必须进行隐藏，由于大多数恶意软件尝试使用名称来进行检索，故而重命名也是一个可选的方式。
- 隐藏 API 记录模块。API 记录模块是大多数沙盒中行为记录的核心组件。它通过将自身注入恶意软件进程中来 Hook 恶意软件的 Win32 API，并在调用时进行记录。恶意

软件会在内存空间中检索此类模块的存在，将其作为识别沙盒的一个特征。在自研 API 记录模块时，一定要注意隐蔽性，要清除任何容易识别模块存在的内存结构。部分恶意软件还会确定 API 中是否存在 Hook 行为，更隐蔽的隐藏需要使用诸如二进制插桩（第 25 章）等技术来实现，借此应对恶意软件的对抗技术。

- 目录与文件名随机化。恶意软件会通过检索已知的控制模块与其他分析工具所使用的目录的命名结构与文件名，来识别是否运行在分析环境中。为了应对此类对抗技术，可以将沙盒控制模块、监控模块以及在沙盒内使用的其他工具所在的目录与对应文件的名称做随机化处理。

主机与沙盒控制模块

沙盒存在互相通信的两个控制模块，支持下发样本文件在沙盒虚拟机中运行，并将运行后的分析结果回传。如图 24- 4 所示，一个控制模块位于每个沙盒虚拟机中，另一个位于沙盒虚拟机外或者宿主机上，分别称之为沙盒控制模块和主机控制模块。

主机与沙盒控制模块可以通过各种编程语言实现，可以使用 C、Python 或者 Go 语言，也可以组合使用。例如，主机控制模块用 Python 编写，沙盒控制模块用 C 编写。接下来将会介绍各个控制模块的工作流程。

主机控制模块

主机控制模块并不只是承担将文件下发到沙盒虚拟机进行分析的责任，还有更多的功能需要其进行实现。主机控制模块常见的工作流程如下所示：

1. 主机控制模块启动时，需要确保集群中的所有沙盒虚拟机都已经启动并且处于可用状态。沙盒虚拟机可能会由于各种原因而出现挂起、崩溃等异常情况，主机控制模块负

责确保沙盒虚拟机始终能够正常运行。

2. 有时候，主机控制模块要负责为沙盒虚拟机创建并维护基础快照。主机控制模块在启动所有沙盒虚拟机后，要为它们创建基础快照或者确保沙盒虚拟机恢复至基础快照。
3. 接收到要分析的样本文件后，主机控制模块会对其进行静态分析，以确定执行所需的操作系统与相关环境。例如，如果接收到的是 Windows PE 可执行文件，主机控制模块会调用 Windows 沙盒虚拟机来运行该文件。对于 Linux 和 macOS 操作系统也与之类似，如图 24- 5 所示。
4. 确定所需的沙盒虚拟机类型后，主机控制模块会在集群中等待一个空闲的沙盒虚拟机。一旦分配到可用的沙盒虚拟机，主机控制模块会将其恢复到基础快照并保持启动状态。沙盒虚拟机启动后，主机控制模块会与沙盒虚拟机内部的沙盒控制模块建立通信，随后下发文件给沙盒控制模块来执行。跟随样本文件一同下发的，还有有关样本文件执行的各种参数与要求。例如，几乎所有沙盒都会为样本执行设置时限，这样样本文件就不会永远执行下去。主机控制模块可以将执行时限传递给沙盒控制模块，沙盒控制模块就会在达到时限后终止样本执行并将分析结果回传主机控制模块。
5. 将样本文件提交给沙盒虚拟机执行，主机控制模块在等待分析完成后，就可以从沙盒虚拟机中获得相关分析数据。

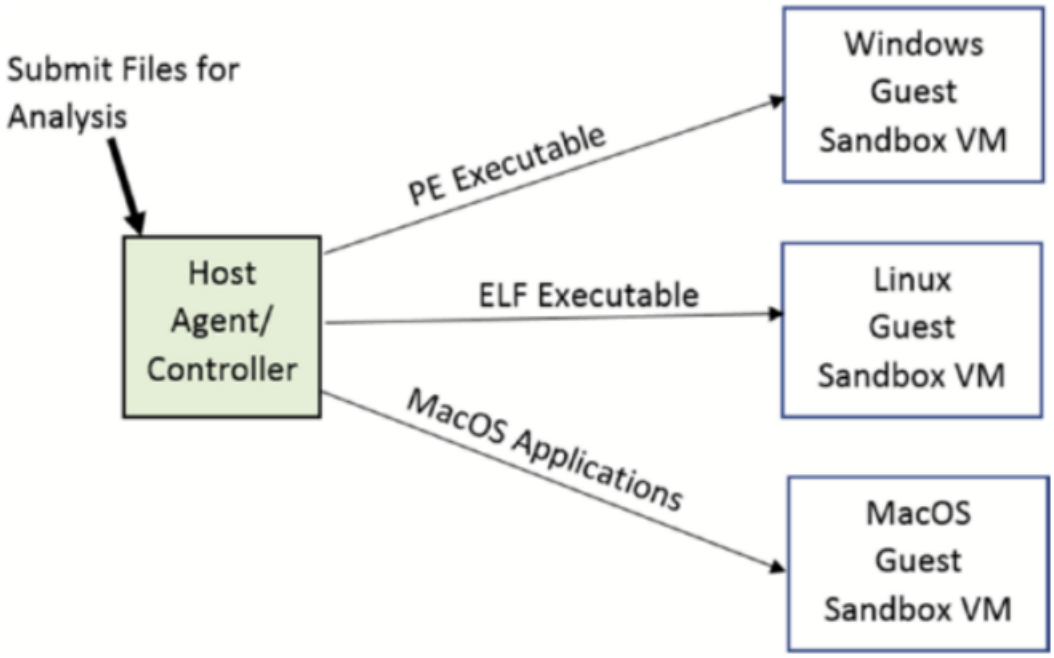

图 24- 5

沙盒控制模块

书接上回,沙盒控制模块在沙盒虚拟机中接收到下发的样本文件,后续的处理流程如下所示。

1. 沙盒控制模块根据文件的类型以及用户自定义的条件与参数，构建分析环境。例如在需要内核监控模块的情况下，确保其能够正常运行。主机控制模块在配置沙盒虚拟机时，通常会覆盖很多操作系统配置项，这些配置也会被包含在基础快照内。因此，沙盒控制模块并不是每次都需要设置操作系统环境。例如在需要插入和设置内核监控模块时，就需要对分析环境进行配置。
2. 在沙盒虚拟机与相关环境根据样本文件类型和所需执行条件配置完成后，沙盒控制模块将会执行样本文件。为了执行各种样本文件，沙盒控制模块通常会使用辅助程序，这些程序通常负责执行样本文件并将其他模块（如监控模块）注入到样本进程中。例如，如果样本文件为可执行文件，辅助程序可以使用在第 10 章中介绍的任意一种代码注入技术来完成运行样本并将监控模块注入进程的任务。如果样本文件为 Word 文档

文件，辅助程序需要使用 Microsoft Word 应用程序打开文件并启动应用程序中的各种调试分析配置，再将监控模块注入到 Microsoft Word 的进程中。该过程如图 24- 6 所示。

3. 随着样本文件运行，各种监控模块和其他模块也在监控样本文件的各种行为或事件，而沙盒控制模块一直要等待样本完成执行。如果样本没有自行终止，沙盒控制模块通常会使用默认时限或者用户提供的时限来终止对样本的监控。尽管不能永无止境地运行和监控样本文件，但超时时限并没有统一标准。沙盒控制模块可以使用 10 秒、15 秒、30 秒、1 分钟、5 分钟或者其他自行指定的时长作为超时时限，该值可以根据文件类型或者其他条件而有所不同，并且可以由提交样本文件进行分析的用户设置的值覆盖。

4. 针对样本的监控完成后，沙盒控制模块会从监控模块以及其他模块处收集相关数据。沙盒控制模块会将收集到所有数据回传给主机控制模块，然后主机控制模块再将其上报给提交样本文件的用户。

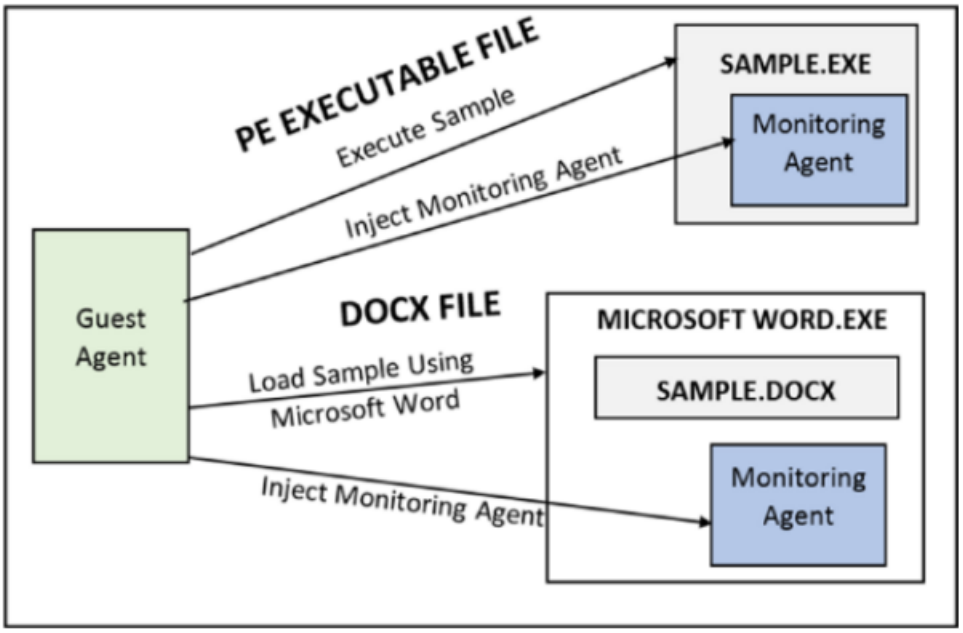

图 24- 6

监控模块

如前所述，监控模块是需要注入到监控的样本文件的进程空间中的组件。该模块通常是一个 DLL 库文件，使用第 10 章中介绍的任一 DLL 注入技术注入到样本文件的进程中。沙盒控制模块通常需要辅助程序实现监控模块 DLL 文件的注入。例如，样本文件是 PE 可执行文件的情况下，辅助程序首先以 SUSPENDED 状态执行样本文件，在注入监控模块 DLL 文件后再恢复挂起的进程（详情请参阅第 10 章中介绍的代码注入原理）。监控模块可以包含各种功能，典型的代表是 API 记录模块与内存转储。

API 记录模块

API 记录模块的工作原理是 Hook 待分析进程使用的所有 Win32 API，并在进程调用时对其进行记录，该模块会记录 Win32 API 的名称以及传递给 API 的所有参数。API 记录模块的典型代表是 APIMiner 与 Cuckoo Monitor。

内存转储与动态脱壳

大多数恶意软件都是加壳的，而运行时通常会在内存中自行脱壳。脱壳后包含了与恶意软件有关的大量信息，这些信息不仅可以帮助分析人员识别该样本文件是否为恶意样本，还能够对其进行分类。在第 13 章与第 15 章中已经探讨了使用内存分析恶意软件的技术，类似的技术也被沙盒系统监控模块中的内存转储模块所使用。内存转储模块通常与 API 记录模块配合使用，使得沙盒可以在执行的各个阶段对样本内存进行转储。如果在样本执行的正确时间点获取内存转储，则会包含恶意软件脱壳后的内容。再将内存转储回传主机控制模块，就可以使用 Yara 等签名对其进行分析。

内核监控模块

之前介绍的监控模块 DLL 文件都位于用户态，例如 API 记录模块是通过 Hook 用户空间的 Win32 API 实现的功能。但如果恶意软件检测到它正在被其他人进行 Hook 或者监视，对抗技术可能会使行为日志极少或者不足以表明其恶意性。

如果没有获得足够多的日志，许多沙盒会通过使用内核监控模块重新分析样本文件。该模块会记录相关进程表现出的行为，包括进程或线程创建相关事件、文件相关事件、网络相关事件和注册表相关事件。尽管这些都是高级别的事件信息，并不像通过用户空间 API 记录模块获取的 Win32 API 使用日志那样细致，但这毕竟是识别恶意软件样本在执行时表现出的恶意行为的最后手段。

ProcMon 与 ETW

不是必须依靠 API 记录模块才能获取有关样本的行为信息，就像可以使用内核监控模块替代用户态工具获得正在运行的进程行为事件。例如，可以使用 ProcMon 来记录样本执行的事件，也可以使用名为 ETW 的 Windows 事件跟踪技术来记录进程的行为事件。但需要注意的是，ProcMon 与 ETW 记录的都是高级别的行为信息，并不像通过 API 记录模块获取 Win32 API 使用日志那样细致且可描述。

欺骗与其他模块

沙盒会使用各种技术来监控样本文件是否存在恶意行为，例如欺骗技术。欺骗技术的典型代表是 Honey File 与 HoneyProcs，通过在沙盒虚拟机中部署诱饵文件与诱饵进程实现监控，详情请参见第 22 章。如果在沙盒内执行的样本文件访问了这些诱饵文件或者诱饵进程，则可能表明该样本文件是恶意的。

主机与沙盒之间的通信信道

出于各种原因，主机控制模块与沙盒控制模块都需要进行通信，大部分情况如下所示：

- 主机控制模块将样本文件提交给沙盒控制模块进行分析。
- 主机控制模块向沙盒控制模块配置各种设置选项与样本执行方式，包括执行样本所需的参数。
- 沙盒控制模块向主机控制模块返回分析结果日志以及在监控过程中收集的各种数据。

主机控制模块与沙盒控制模块间的所有通信通常都是通过二者间建立的网络连接进行的。大多数沙盒虚拟机都会启用网络，沙盒控制模块可以在特定端口上监听传入连接。主机控制模块通过该监听端口连接到沙盒控制模块，与其建立双向通信信道交换数据。

主机控制模块与沙盒控制模块间也可以使用其他通信方式，例如通过虚拟机管理程序（hypervisor）在沙盒虚拟机内部打开串行端口，但最常用的方式仍然是基于 TCP/IP 的网络套接字通信。

通过文件或流进行记录

在监控模块对应的章节中，介绍模块在沙盒虚拟机内如何记录各种行为信息，如 API 日志、事件与内存转储。这些数据需要回传给主机控制模块以供进一步分析，但监控模块与其他模块是如何在沙盒虚拟机中记录这些数据的呢？

主要有两种方式：

- 将所有日志、事件与内存转储都写入磁盘文件，沙盒控制模块通过建立的通信信道将其传输给主机控制模块。
- 监控模块与其他模块不将相关数据写入磁盘文件，而是通过某种进程间通信技术将数据直接传输给沙盒控制模块，随后沙盒控制模块将其传输到主机控制模块。

处理文件相对容易，所以第一种方式更为简单。但该方式存在显著的缺点，如果某些恶意软件检索此类文件并将其删除，就可以将行为痕迹擦除。此外，特定类型的恶意软件（如勒索软件）可能会将这些日志文件加密或者执行其他存在影响日志文件的操作，从而破坏了数据的可用性。为了应对这种不足，很多沙盒都会采用第二种方式，由监控模块与沙盒控制模块建立进程间通信，从而避免恶意软件破坏日志文件。

基于沙盒分析结果编写检测规则

前文已经介绍了沙盒如何运行样本文件、监控并记录相关行为与事件，最后将分析结果、内存转储和其他数据都返回给提交样本文件的使用者。

得到分析日志后，使用者就需要基于此判断样本文件是否是恶意文件。本书之前的章节使用APIMiner 来记录样本文件调用的 API，分析人员检查生成的 API 调用日志以发现与恶意行为相关的 API 调用序列。在沙盒 API 日志中也可以使用相似的方式进行分析，首先根据恶意软件的 API 调用序列生成签名，这些签名就可以用于从沙盒 API 日志来识别并分类恶意软件。

如第 10 章介绍的，远程进程执行相关的一系列 API（如 CreateProcess、VirtualAllocEx 与 WriteProcessMemory）都指向代码注入这一恶意行为。将此类的 API 调用序列转换为签名，应用于沙盒输出的 API 日志，就可以轻松发现使用相同 API 调用序列进行代码注入的恶意软件。其他类型的恶意 API 调用序列也可以使用相同的方式处理。

类似的，沙盒中的内存转储模块可以在样本执行的各个阶段获取进程内存。随后分析提取的内存中是否存在恶意字符串，来判断样本文件是否是恶意文件，甚至对恶意样本进行分类。如第 22 章所示，分析人员可以针对内存转储编写 Yara 规则实现检测。

将沙盒与机器学习算法结合

机器学习技术已经深度与现代软件融合，网络安全行业也在积极利用该技术，例如与沙盒相结合构建威胁检测模型。首先，大量已标记的、常见的恶意软件样本文件被送入沙盒执行获取 API 日志。其次，基于 API 日志提取相关特征，特征可以是调用的 API、API 调用的顺序以及 API 调用的参数等。最后，将这些特征输入各种机器学习算法以构建基线模型，然后实际部署这些模型即可检测恶意软件。当检测产品接收到新样本文件进行分析时，通过相同的沙盒运行获取 API 日志再提取特征。利用之前构建并部署的基线模型，根据提取的特征判断样本是良性的还是恶意的。

目前，机器学习模型尚不能以百分百的准确性判断样本是良性的还是恶意的。与其他各种检测技术一样，机器学习也存在误报与漏报的情况。机器学习模型识别恶意样本的有效性其实并不全取决于算法，而是取决于从样本文件中提取的特征。如果沙盒能够正确地执行与分析样本文件的行为、提取样本的所有 API 调用，就可以帮助我们构建更好的特征。

正是从样本文件中获取的特征帮助机器学习算法区分良性文件与恶意文件。获取的特征越差，检测率就越低，故而重点应该放在特征提取上。目前用来构建机器学习模型的算法很好，但无论付出多大的努力来改进这些模型，总会存在一些不可避免的误报与漏报。

这也是将多种检测技术相结合的重要原因。例如将内存转储与 Yara 签名相结合的检测，可以提高整体的准确率和效率。同样的，也可以与反病毒引擎相结合。另外，数字签名等静态属性信息也可以对判断样本是否为恶意文件提供帮助。还可以组合使用与网络有关的检测引擎，进一步提供更多的上下文信息。多种检测方式可以无缝集成在一起，协同提供高精度的检测结果。

总结

沙盒已经成为当今几乎所有与恶意软件相关的检测产品的重要组成部分。本章中介绍了沙盒是什么，以及为什么沙盒能够成为具有重要价值的的检测技术。此外，也描述了如何将沙盒集成到各种检测产品中，例如网络安全产品、端点安全产品以及电子邮件安全产品。本章将从样本文件提交到沙盒进行分析到处理数据结果的全流程进行了介绍，甚至包括沙盒的设计架构与各种构成组件（例如 API 记录模块与其他行为监控模块，利用这些监控模块可以观察并记录样本文件表现出的各种行为）。最后介绍了如何对沙盒返回的数据进行检测，以及如何与机器学习算法相结合来改进自动化并加速检测。