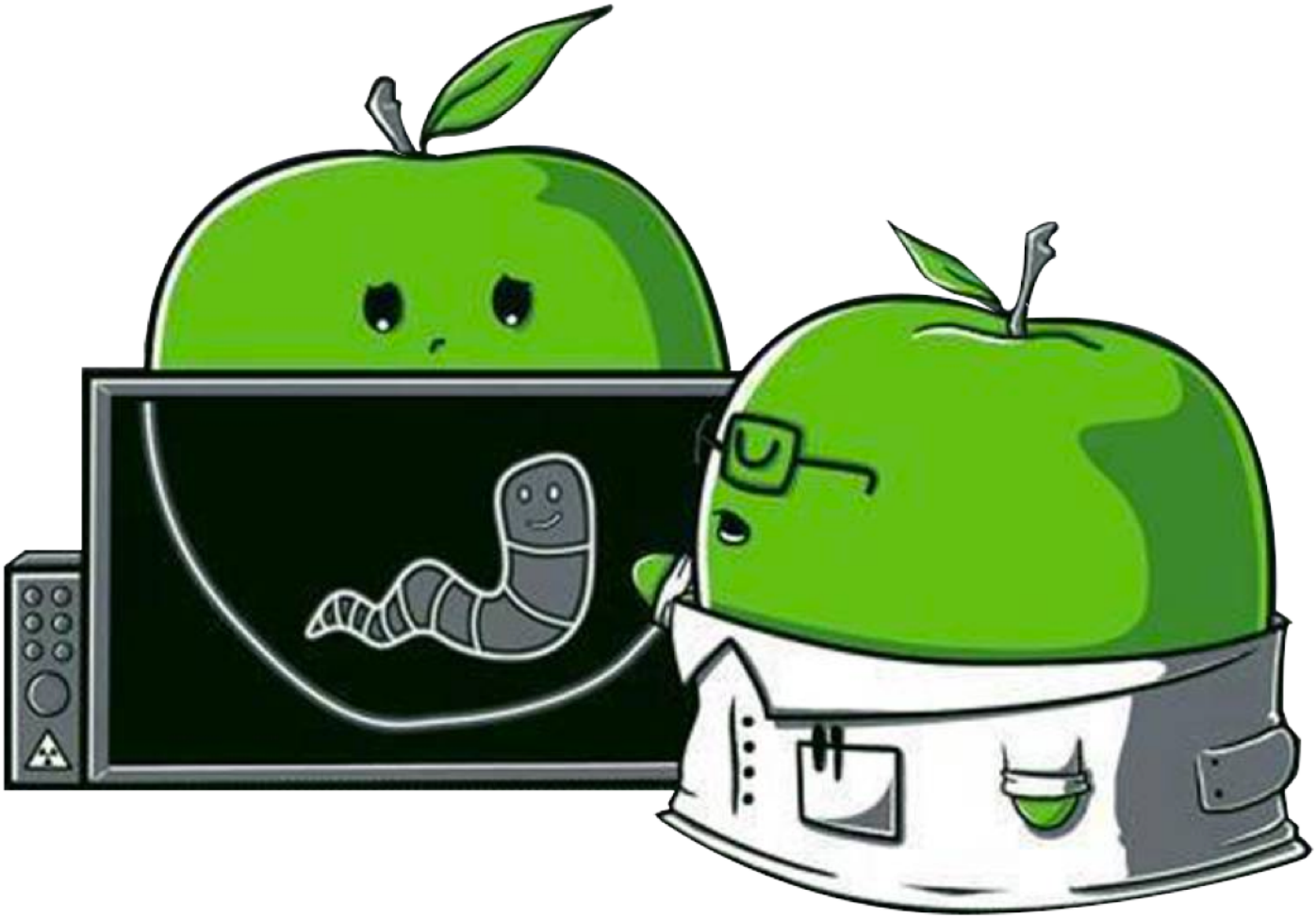


第0x9章：动态监控（工具）

笔记:

这本书正在进行中。

我们鼓励您直接在这些页面上发表评论 ...建议编辑、更正和/或其他内容！

要发表评论，只需突出显示任何内容，然后单击 出现在屏幕上的图标
就在文档的边界上）。

笔记:

由于动态分析涉及执行恶意软件（以观察其行为），因此始终在虚拟机（VM）或专用恶意软件分析机上执行此类分析。

...换句话说，不要在主（基本）系统上执行动态分析！

在本章中，我们将重点介绍各种动态分析监控工具。具体来说，我们将说明进程、文件和网络监视器如何有效地提供对恶意软件样本的功能和能力的宝贵见解。

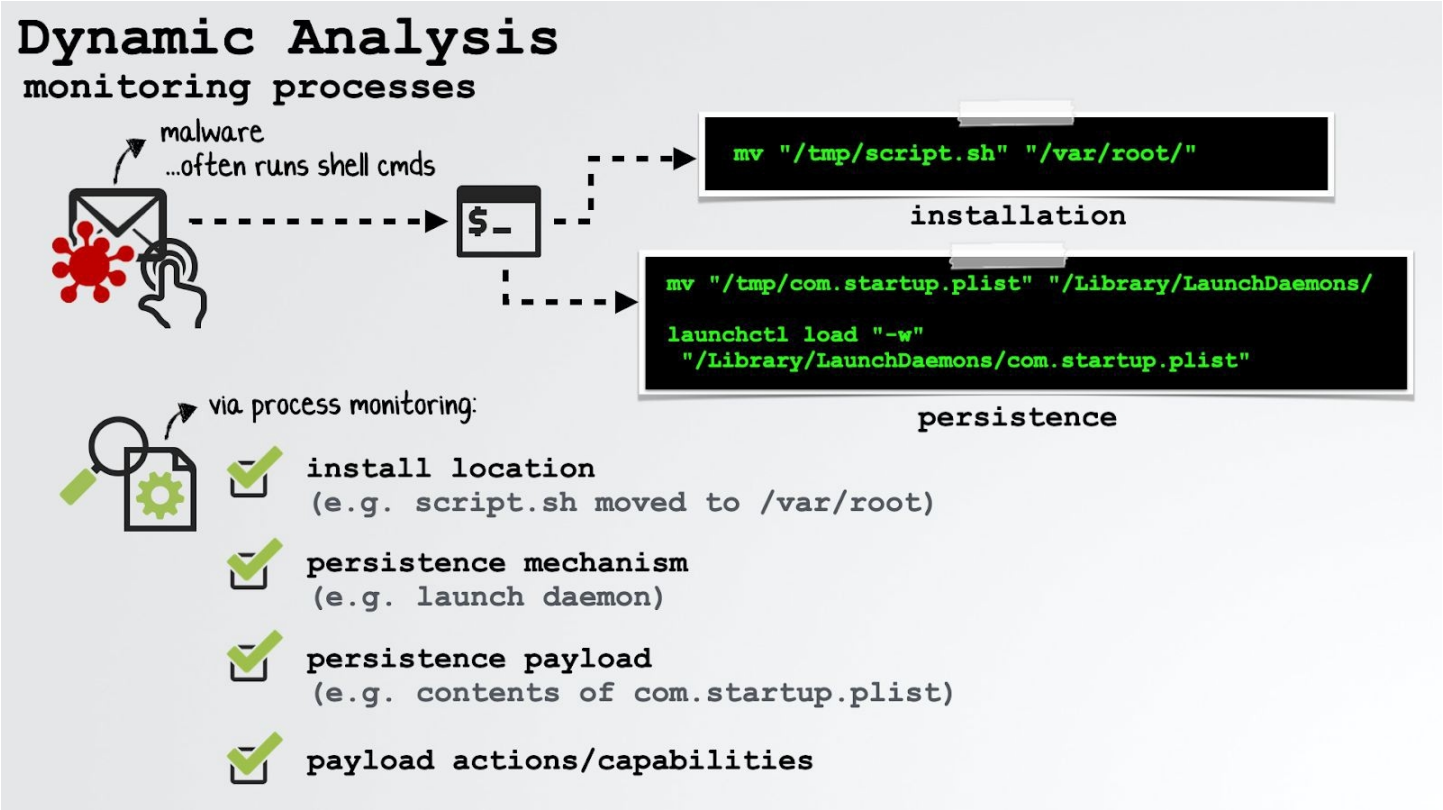
过程监控

恶意软件通常会产生或执行子进程。如果通过过程监视器进行观察，这些过程可能会迅速提供对恶意软件行为和功能的洞察。

通常，此类进程是内置的（系统）命令行实用程序，恶意软件执行这些程序是为了（惰性地）委托所需的操作。

例如:

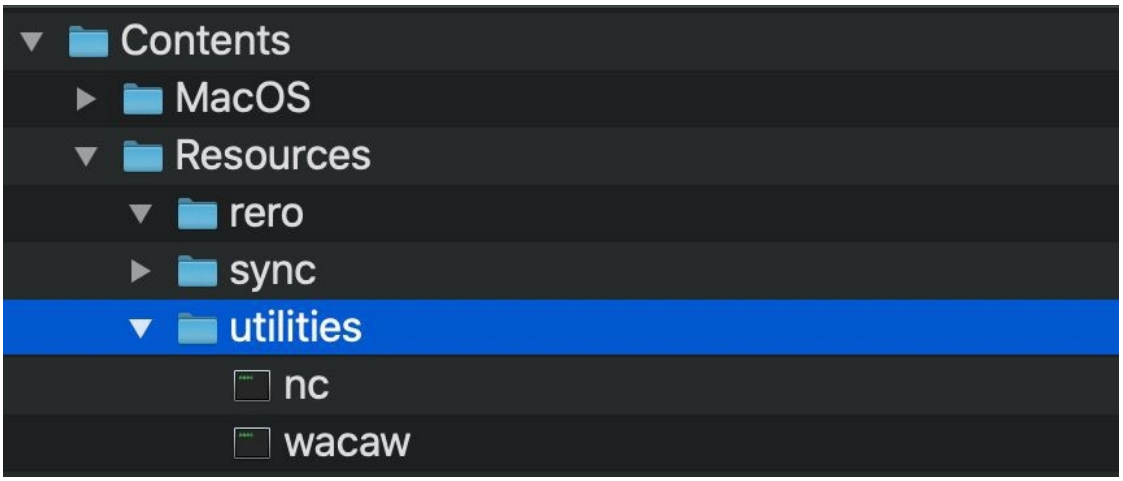
- 恶意安装程序可能会调用move（/bin/mv）或copy（/bin/cp）实用程序来持续安装恶意软件。
- 为了调查系统，恶意软件可能会调用进程状态（/bin/ps）实用程序来获取正在运行的进程列表，或者调用/usr/bin/whoami实用程序来确定当前用户的权限。
- 然后，可以通过/usr/bin/curl将此调查的结果导出到远程命令和控制服务器。



在上图中，进程监视器会快速显示恶意样本的安装逻辑（将script.sh从临时位置复制到/var/root），以及其持久性机制（启动守护程序：com.startup.plist） ...不需要静态分析！

恶意软件还可能产生与原始恶意软件样本打包在一起或从远程命令和控制服务器下载的其他二进制文件。

例如，OSX。Eleanor [1]部署了几个实用程序来扩展恶意软件的功能。具体来说，它与著名的网络实用程序nc（netcat）和wacaw预先捆绑在一起，wacaw是“Mac OS X的命令行工具，允许从连接的摄像头中捕获静止图片和视频” [2]。



奥斯。埃莉诺的预捆绑公用设施

通过进程监视器，我们可以观察执行这些打包实用程序的恶意软件，这反过来又允许我们被动地确定恶意软件的功能（即，能够通过网络摄像头记录受感染系统的用户）。

> 笔记:
>
> 用OSX打包的二进制文件。埃莉诺本身并不恶毒。
>
> 相反，这些实用程序只是提供了恶意软件作者想要整合到恶意软件中的功能（例如网络摄像头录制），但可能太懒了，无法自行编写。

另一个嵌入二进制文件的恶意软件样本是OSX。果蝇：

> “[恶意软件]包含一个编码的Mach-O二进制文件，该文件被写入/tmp/客户。在通过调用chmod使这个二进制文件可执行之后，子例程通过调用open2来派生一个子进程，以执行[binary]。” [3]

奥斯。果蝇是用Perl编写的，这限制了它执行“低级”操作的能力，例如在macOS上生成合成鼠标和键盘事件。为了解决这个缺点，恶意软件作者加入了一个能够执行这些额外功能的嵌入式Mach-O二进制文件。

如前所述，流程监视器可以被动地观察流程的执行情况，显示生成流程的流程标识符和路径。更全面的流程监控器可以提供附加信息，例如传递给子流程的流程层次结构（即祖先）流程参数，以及新流程的代码签名信息

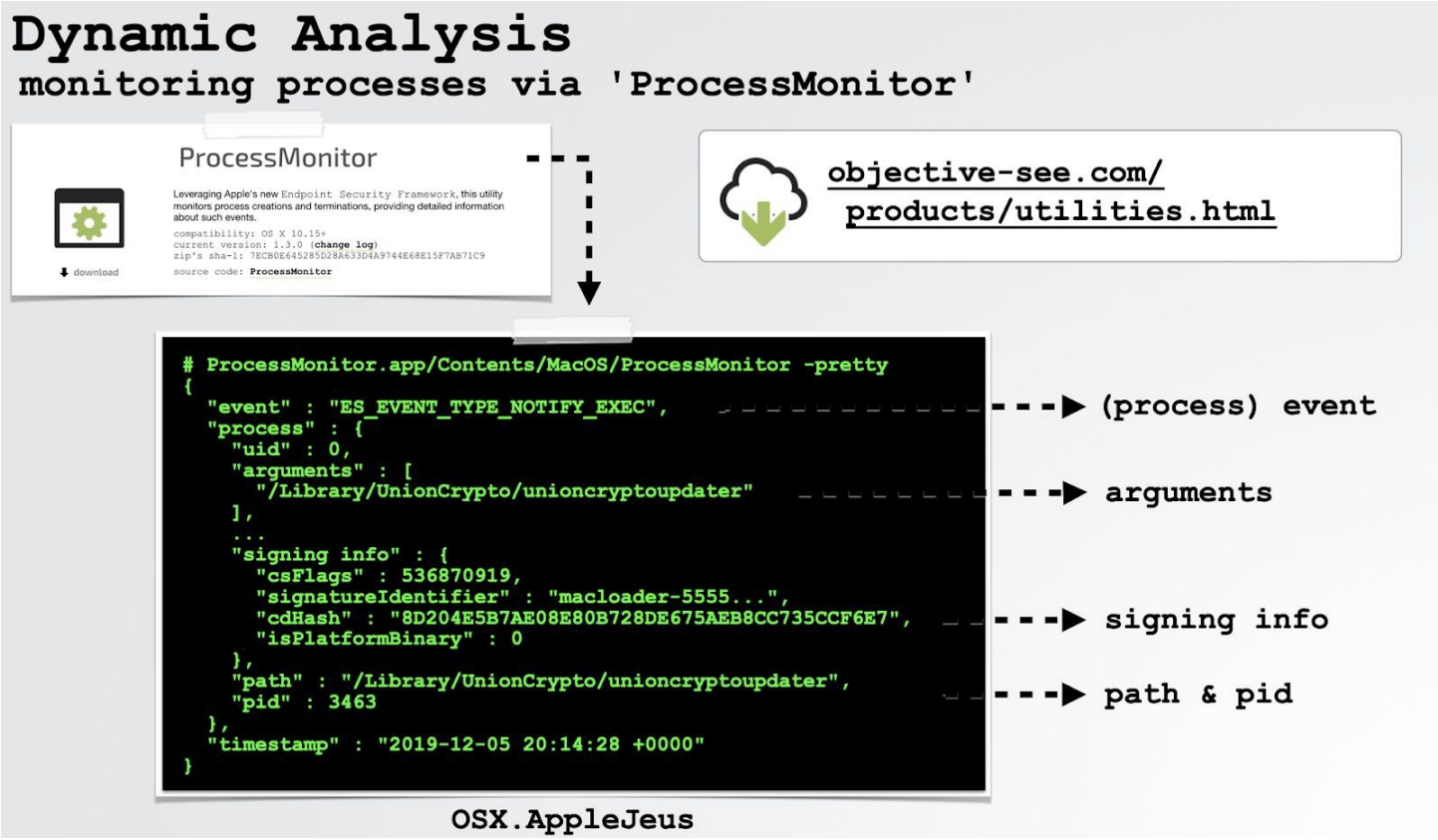
创建（子）进程。在这些附加信息中，进程参数尤其有价值，因为它们可以揭示恶意软件正在授权的操作。

不幸的是，macOS没有提供功能完整的内置过程监控实用程序。

> 笔记:
>
> 如果使用-f exec命令行标志调用，Apple的fs_usage实用程序将捕获并显示流程事件的子集。
>
> 然而，由于它不能全面捕获所有进程事件，也不能显示进程参数等基本信息，因此它对于恶意软件分析目的并不特别有用。（即，$ open Calculator.app不会导致生成计算器的事件报告）

然而，开源的“ProcessMonitor” [4]实用程序是（由yours Really创建的）专门为方便Mac恶意软件的动态分析而创建的。

> 笔记:
>
> 要确保ProcessMonitor能够运行，必须满足几个（苹果利用的）先决条件，包括:
>
> 1. 向终端授予“完全磁盘访问权”。应用程序
> 2. 以root用户身份运行ProcessMonitor
> 3. 指定ProcessMonitor二进制文件的完整路径
>
> 有关更多信息，请参阅:
>
> ProcessMonitor的文档



过程监视器[5]

如上图所示，ProcessMonitor将显示进程事件（exec、fork、exit等）以及进程：

- 用户id（uid）
- 命令行参数
- （已报道）代码签名信息
- 完整路径
- 过程标识符（pid）

ProcessMonitor还报告计算出的代码签名信息（包括签名权限）、父pid和完整的流程层次结构。下面的示例说明了这一点，其中我们使用-lart命令行参数执行ls命令：

```
#进程监视器。app/Contents/MacOS/ProcessMonitor -pretty
{
  “事件”：“事件类型通知执行”，
  “过程”：{
    “签名信息（计算）”：{
```

```
      "signatureID" : "com.apple.ls",
      "signatureStatus" : 0,
      "signatureSigner" : "Apple",
      "signatureAuthorities" : [
        “软件签名”，
        “苹果代码签名认证机构”、“苹果根CA”
      ]
    },
    "uid" : 501,
    “论据” : [
      "ls",
      "-lart"
    ],
    "ppid" : 3051,
    “祖先”：[ 3051,
      3050,
      447,
      1
    ],
    “路径”：“/bin/ls”，
    “签名信息（已报告）”：{“teamID”
      ：（空）”，“csFlags”：
      604009233,“signingID”：
      “com.apple.ls”，
      “platformBinary”：1,
      "cdHash" : "5467482A6DEBC7A62609B98592EAE3FB35964923"
    },
    "pid" : 7482
  },
  “时间戳”：“2020-01-26 22:50:12+0000”
}
```

现在，让我们简单地看一下ProcessMonitor的输出，它被动地观察Lazarus（APT）组安装程序生成的进程[5]:

```
#进程监视器。app/Contents/MacOS/ProcessMonitor -pretty

{
  “事件”：“事件类型通知执行”，
  “过程”：{
    "uid" : 0,
```

```
    “论据” : [
      "mv",
      "/Applications/UnionCryptoTrader.app/Contents/
                    Resources/.vip.unioncrypto.plist",
      "/Library/LaunchDaemons/vip.unioncrypto.plist"
    ],
    "ppid" : 3457,
    “祖先”：[ 3457,
      951,
      1
    ],
    “签名信息”：{“csFlags”：
      603996161,
      "signatureIdentifier" : "com.apple.mv",
      "cdHash" : "7F1F3DE78B1E86A622F0B07F766ACF2387EFDCD",
      "isPlatformBinary" : 1
    },
    “路径”：“/bin/mv”，
    “pid”：3458
  },
  “时间戳”：“2019-12-05 20:14:28+0000”
}

...

{
  “事件”：“事件类型通知执行”，
  “过程”：{
    "uid" : 0,
    “论据” : [
      "mv",
      "/Applications/UnionCryptoTrader.app/Contents/Resources/.unioncryptoupdater",
      "/Library/UnionCrypto/unioncryptoupdater"
    ],
    "ppid" : 3457,
    “祖先”：[ 3457,
      951,
      1
    ],
    “签名信息”：{“csFlags”：
      603996161,
      "signatureIdentifier" : "com.apple.mv",
      "cdHash" : "7F1F3DE78B1E86A622F0B07F766ACF2387EFDCD",
```

```
      "isPlatformBinary" : 1
    },
    “路径”：“/bin/mv”，
    “pid”：3461
  },
  “时间戳”：“2019-12-05 20:14:28+0000”
}

...

{
  “事件”：“事件类型通知执行”，
  “过程”：{
    "uid" : 0,
    “论点”：[ "/Library/UnionCrypto/unioncryptoupdater"
    ],
    "ppid" : 1,
    “祖先”：[1]
    ],
    “签名信息”：{“csFlags”：
      536870919,
      "signatureIdentifier" : "macloader-55554944ee2cb96a1f5132ce8788c3fe0dfe7392",
      "cdHash" : "8D204E5B7AE08E80B728DE675AEB8CC735CCF6E7",
      "isPlatformBinary" : 0
    },
    “路径”：“/Library/UnionCrypto/unioncryptoupdater”，“pid”
    ：3463
  },
  “时间戳”：“2019-12-05 20:14:28+0000”
}
```


从这个输出（特别是进程及其参数），我们观察到恶意安装程序:

1. 执行内置的`/bin/mv`实用程序，将隐藏的属性列表（`.vip.unioncrypto.plist`）从安装程序的资源/目录移动到`/Library/LaunchDaemons`.

2. 执行`/bin/mv`将隐藏的二进制文件（`.unioncryptoupdater`）从安装程序的`Resources/`目录移动到`/Library/UnionCrypto/`。

3. 启动此二进制文件（/Library/UnionCrypto/unioncryptoupdater）

通过这些过程观察，我们可以快速准确地了解恶意软件是如何持久存在的（启动守护进程），并识别恶意软件的持久组件（unioncryptoupdater二进制文件）。

这可以通过对安装程序脚本的静态分析或手动检查启动守护程序plist、vip来确认。UnionCrypto。plist（正如预期的那样，它引用了/Library/UnionCrypto/unioncryptoupdater二进制文件）：

```
# cat /Library/LaunchDaemons/vip。unioncrypto。普利斯特

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" ...>
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>vip.unioncrypto.product</string>
    <key>ProgramArguments</key>
    <array>
            <string>/Library/UnionCrypto/unioncryptoupdater</string>
   </array>
   <key>RunAtLoad</key>
   <true/>
</dict>
</plist>
```

过程监控还可以揭示恶意样本的核心功能。例如，OSX。WindTail[6]的主要目的是收集并过滤受感染系统中的文件。虽然这可以通过静态分析方法确定，比如分解恶意软件的二进制文件，但也可以通过过程监视器进行观察。
具体地说，如ProcessMonitor的以下简略输出所示，我们可以观察到恶意软件首先创建要收集的文件（psk.txt）的zip存档，然后通过curl命令进行过滤：

```
#进程监视器。app/Contents/MacOS/ProcessMonitor -pretty

{
  “事件”：“事件类型通知执行”，
  “过程”：{
```