WIPO
WORLD
INTELLECTUAL PROPERTY
ORGANIZATION

# DOCUMENT MADE AVAILABLE UNDER THE PATENT COOPERATION TREATY (PCT)

| | | |
|---|---|---|
| International application number: | | **PCT/CN2018/081303** |
| International filing date: | | **30 March 2018 (30.03.2018)** |
| Document type: | | **Certified copy of priority document** |
| Document details: | Country/Office: | **CN** |
| | Number: | **201710805670.4** |
| | Filing date: | **08 September 2017 (08.09.2017)** |
| Date of receipt at the International Bureau: | | **10 May 2018 (10.05.2018)** |

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a),(b) or (b-*bis*)

34, chemin des Colombettes
1211 Geneva 20, Switzerland
www.wipo.int

# 证　明

本证明之附件是向本局提交的下列专利申请文件副本。

申　　请　　号：201710805670.4

申　请　类　型：发明专利

发明 创造 名称：IOS平台隐藏dylib文件的方法、存储介质、电子设备及系统

申　　请　　日：2017.09.08

申　　请　　人：武汉斗鱼网络科技有限公司

发明人或设计人：周志刚、陈少杰、张文明

局长

申长雨

2018 年 05 月 08 日

2017108056704

# 权 利 要 求 书

1、一种 IOS 平台隐藏 dylib 文件的方法，其特征在于，包括：

S1：获取 IOS 应用内的所有 Mach-o 文件及每个 Mach-o 文件内的 dylib 文件名称获取函数；

S2：编写一 hook 函数，所述 hook 函数用于获取 dylib 文件名称获取函数获取的 dylib 文件的名称；

S3：当 IOS 应用调用 dylib 文件名称获取函数进行 IOS 应用内所有 dylib 文件名称的获取时，通过 hook 函数获取 dylib 文件名称获取函数获取的 dylib 文件名称，根据 dylib 文件名称判断是否含有待隐藏 dylib 文件，若是，转到 S4，若否，转到 S5；

S4：隐藏待隐藏 dylib 文件名称，并将隐藏了待隐藏 dylib 文件名称后的 dylib 文件名称数据返回给 IOS 应用；

S5：将 dylib 文件名称获取函数取的 dylib 文件名称返回给 IOS 应用。

2、如权利要求 1 所述的一种 IOS 平台隐藏 dylib 文件的方法，其特征在于，对于 Mach-o 文件内 dylib 文件名称获取函数的获取，具体为：

S101： 获取 IOS 应用内的所有 Mach-o 文件及每个 Mach-o 文件所对应的内存起始地址，并根据内存起始地址获取 Mach-o 文件在内存中的虚拟内存地址；

S102：根据 Mach-o 文件在内存中的虚拟内存地址，解析 Mach-o 文件，得到每个 Mach-o 文件内每个函数的名称和指针；

S103：遍历得到的每个 Mach-o 文件内每个函数的名称，找到每个 Mach-o 文件内的 dylib 文件名称获取函数。

3、如权利要求 1 所述的一种 IOS 平台隐藏 dylib 文件的方法，

其特征在于：

所述 hook 函数用于将目标函数的内存地址进行替换成设定地址；

当目标函数执行时，跳转至设定地址以进行 dylib 文件名称获取函数获取的 dylib 文件名称的获取；

所述目标函数为 dylib 文件名称获取函数。

4、如权利要求 1 所述的一种 IOS 平台隐藏 dylib 文件的方法，其特征在于：

所述 IOS 应用还用于调用 Mach-o 文件内的 dylib 文件个数获取函数获取 IOS 应用内的 dylib 文件个数；

当 dylib 文件名称获取函数获取的 dylib 文件名称中含有待隐藏 dylib 文件名称，则返回给 IOS 应用的 dylib 文件个数为获取的 dylib 文件总个数减去待隐藏 dylib 文件的个数；

当 dylib 文件名称获取函数获取的 dylib 文件名称中不含有待隐藏 dylib 文件名称，则返回给 IOS 应用的 dylib 文件个数即为获取的 dylib 文件总个数。

5、如权利要求 4 所述的一种 IOS 平台隐藏 dylib 文件的方法，其特征在于：通过 hook 方式对返回给 IOS 应用的 dylib 文件的个数进行修改。

6、一种存储介质，该存储介质上存储有计算机程序，其特征在于：所述计算机程序被处理器执行时实现权利要求 1 至 5 任一项所述的方法。

7、一种电子设备，包括存储器和处理器，存储器上储存有在处理器上运行的计算机程序，其特征在于：所述处理器执行所述计算机程序时实现权利要求 1 至 5 任一项所述的方法。

2017108056704

8、一种 IOS 平台隐藏 dylib 文件的系统，其特征在于，包括：

获取模块，其用于获取 IOS 应用内的所有 Mach-o 文件及每个 Mach-o 文件内的 dylib 文件名称获取函数；

编写模块，其用于编写一 hook 函数，所述 hook 函数用于获取 dylib 文件名称获取函数获取的 dylib 文件的名称；

判断与执行模块，其用于当 IOS 应用调用 dylib 文件名称获取函数进行 IOS 应用内 dylib 文件名称的获取时，通过 hook 函数获取 dylib 文件名称获取函数获取的 dylib 文件名称，根据 dylib 文件名称判断是否含有待隐藏 dylib 文件，若是，则隐藏待隐藏 dylib 文件名称，并将隐藏了待隐藏 dylib 文件名称后的 dylib 文件名称数据返回给 IOS 应用，若否，则将 dylib 文件名称获取函数取的 dylib 文件名称返回给 IOS 应用。

9、如权利要求 8 所述的一种 IOS 平台隐藏 dylib 文件的系统，其特征在于：所述获取模块对于 Mach-o 文件内 dylib 文件名称获取函数的获取，具体为：

获取 IOS 应用内的所有 Mach-o 文件及每个 Mach-o 文件所对应的内存起始地址，并根据内存起始地址获取 Mach-o 文件在内存中的虚拟内存地址；

根据 Mach-o 文件在内存中的虚拟内存地址，解析 Mach-o 文件，得到每个 Mach-o 文件内每个函数的名称和指针；

遍历得到的每个 Mach-o 文件内每个函数的名称，找到每个 Mach-o 文件内的 dylib 文件名称获取函数。

10、如权利要求 9 所述的一种 IOS 平台隐藏 dylib 文件的系统，其特征在于：所述 hook 函数用于将目标函数的内存地址进行替换成设定地址；当目标函数执行时，跳转至设定地址以进行 dylib 文件名称获

取函数获取的 dylib 文件名称的获取；所述目标函数为 dylib 文件名称获取函数。

2017108056704

# 说 明 书

## IOS 平台隐藏 dylib 文件的方法、存储介质、电子设备及系统

### 技术领域

本发明涉及信息处理领域，具体涉及一种 IOS 平台隐藏 dylib 文件的方法、存储介质、电子设备及系统。

### 背景技术

当前，随着移动设备的日益普及，移动应用产业尤其是 IOS 应用随之得到飞速发展，且 IOS 应用的开发模式和代码框架也随之发生了巨大变化。对于 IOS 平台，其应用程序的构成分为主程序和模块文件，模块文件即为 dylib 文件，dylib 文件是 IOS 平台上的动态链接库文件。在 IOS 应用开发的过程中，若需要编写某些功能模块注入到对应的进程中运行，则均是编写一个 dylib 文件，然后将该 dylib 文件注入到对应的进程中运行。

在 IOS 应用开发完成并进行运行后，通常会编写一个 dylib 文件注入到运行的 IOS 应用中，进行 IOS 应用的运行状态等相关数据的获取，以便于后期开发人员对 IOS 应用的调试和维护，由于 IOS 系统资源机制的原因，IOS 应用的进程经常会关闭一些 dylib 文件的运行，以节省系统资源，保证系统资源的合理利用，故注入到 IOS 应用中用于获取 IOS 应用运行状态的 dylib 文件会被误禁止，进行影响开发人员对 IOS 应用后期调试工作的正常进行。

### 发明内容

针对现有技术中存在的缺陷，本发明的目的在于提供一种 IOS 平台隐藏 dylib 文件的方法，能够有效保证应用后期调试工作的正常

进行。

为达到以上目的，本发明采取的技术方案是，包括：

S1：获取 IOS 应用内的所有 Mach-o 文件及每个 Mach-o 文件内的 dylib 文件名称获取函数；

S2：编写一 hook 函数，所述 hook 函数用于获取 dylib 文件名称获取函数获取的 dylib 文件的名称；

S3：当 IOS 应用调用 dylib 文件名称获取函数进行 IOS 应用内所有 dylib 文件名称的获取时，通过 hook 函数获取 dylib 文件名称获取函数获取的 dylib 文件名称，根据 dylib 文件名称判断是否含有待隐藏 dylib 文件，若是，转到 S4，若否，转到 S5；

S4：隐藏待隐藏 dylib 文件名称，并将隐藏了待隐藏 dylib 文件名称后的 dylib 文件名称数据返回给 IOS 应用；

S5：将 dylib 文件名称获取函数取的 dylib 文件名称返回给 IOS 应用。

在上述技术方案的基础上，对于 Mach-o 文件内 dylib 文件名称获取函数的获取，具体为：

S101： 获取 IOS 应用内的所有 Mach-o 文件及每个 Mach-o 文件所对应的内存起始地址，并根据内存起始地址获取 Mach-o 文件在内存中的虚拟内存地址；

S102：根据 Mach-o 文件在内存中的虚拟内存地址，解析 Mach-o 文件，得到每个 Mach-o 文件内每个函数的名称和指针；

S103：遍历得到的每个 Mach-o 文件内每个函数的名称，找到每个 Mach-o 文件内的 dylib 文件名称获取函数。

在上述技术方案的基础上，

所述 hook 函数用于将目标函数的内存地址进行替换成设定地址；

当目标函数执行时，跳转至设定地址以进行 dylib 文件名称获取函数获取的 dylib 文件名称的获取；

所述目标函数为 dylib 文件名称获取函数。

在上述技术方案的基础上，

所述 IOS 应用还用于调用 Mach-o 文件内的 dylib 文件个数获取函数获取 IOS 应用内的 dylib 文件个数；

当 dylib 文件名称获取函数获取的 dylib 文件名称中含有待隐藏 dylib 文件名称，则返回给 IOS 应用的 dylib 文件个数为获取的 dylib 文件总个数减去待隐藏 dylib 文件的个数；

当dylib文件名称获取函数获取的dylib文件名称中不含有待隐藏 dylib 文件名称，则返回给 IOS 应用的 dylib 文件个数即为获取的 dylib 文件总个数。

在上述技术方案的基础上，通过 hook 方式对返回给 IOS 应用的 dylib 文件的个数进行修改。

本发明还提供一种存储介质，该存储介质上存储有计算机程序，所述计算机程序被处理器执行时实现上述所述的方法。

本发明还提供一种电子设备，包括存储器和处理器，存储器上储存有在处理器上运行的计算机程序，所述处理器执行所述计算机程序时实现上述所述的方法。

本发明还提供一种 IOS 平台隐藏 dylib 文件的系统，包括：

获取模块，其用于获取 IOS 应用内的所有 Mach-o 文件及每个 Mach-o 文件内的 dylib 文件名称获取函数；

编写模块，其用于编写一 hook 函数，所述 hook 函数用于获取 dylib 文件名称获取函数获取的 dylib 文件的名称；

判断与执行模块，其用于当 IOS 应用调用 dylib 文件名称获取函

数进行 IOS 应用内 dylib 文件名称的获取时，通过 hook 函数获取 dylib 文件名称获取函数获取的 dylib 文件名称，根据名称判断是否含有待隐藏 dylib 文件，若是，则隐藏待隐藏 dylib 文件的名称，并将隐藏了待隐藏 dylib 文件名称后的 dylib 文件名称数据返回给 IOS 应用，若否，则将 dylib 文件名称获取函数取的 dylib 文件名称返回给 IOS 应用。

在上述技术方案的基础上，所述获取模块对于 Mach-o 文件内 dylib 文件名称获取函数的获取，具体为：

获取 IOS 应用内的所有 Mach-o 文件及每个 Mach-o 文件所对应的内存起始地址，并根据内存起始地址获取 Mach-o 文件在内存中的虚拟内存地址；

根据 Mach-o 文件在内存中的虚拟内存地址，解析 Mach-o 文件，得到每个 Mach-o 文件内每个函数的名称和指针；

遍历得到的每个 Mach-o 文件内每个函数的名称，找到每个 Mach-o 文件内的 dylib 文件名称获取函数。

在上述技术方案的基础上，所述 hook 函数用于将目标函数的内存地址进行替换成设定地址；当目标函数执行时，跳转至设定地址以进行 dylib 文件名称获取函数获取的 dylib 文件名称的获取；所述目标函数为 dylib 文件名称获取函数。

与现有技术相比，本发明的优点在于：通过 hook 的方式，当 IOS 应用调用 dylib 文件名称获取函数进行 dylib 文件名称的获取时，通过 hook 函数获取 dylib 文件名称获取函数获取的 dylib 文件名称，隐藏待隐藏的 dylib 文件，即相当于隐藏了应用内的用于获取 IOS 应用运行状态的 dylib 文件，避免 IOS 应用对该类 dylib 文件的禁止运行操作，保证开发人员能够正常获取到 IOS 应用运行状态的数据，保证

IOS 应用调试工作的正常进行。

## 附图说明

图 1 为本发明实施例中 IOS 平台隐藏 dylib 文件的方法的流程图；

图 2 为本发明实施例一种电子设备的结构示意图。

## 具体实施方式

以下结合附图及实施例对本发明作进一步详细说明。

参见图 1 所示，本发明实施例提供一种 IOS 平台隐藏 dylib 文件的方法，用于当 IOS 应用对应用内的 dylib 文件获取时，隐藏应用内的用于获取 IOS 应用运行状态的 dylib 文件，避免 IOS 应用对该类 dylib 文件获取而禁止运行，保证开发人员对 IOS 应用调试工作的正常进行。

本发明实施例中，dylib 文件名称获取函数为 IOS 的系统函数 _dyld_get_image_name，通过该函数可以获取 IOS 应用中加载的每个 dylib 文件的名称，该函数的原型为：

const char*  _dyld_get_image_name(uint32_t image_index);

该函数中，参数 uint32_t image_index 用来标示获取的 dylib 文件的索引，该函数的返回值 const char* 用来标示获取到的 dylib 文件的名称。

本发明实施例的 IOS 平台隐藏 dylib 文件的方法具体包括：

S1：获取 IOS 应用内的所有 Mach-o 文件及每个 Mach-o 文件内的 dylib 文件名称获取函数。对于 Mach-o 文件内 dylib 文件名称获取函数的获取，具体为：

S101： 获取 IOS 应用内的所有 Mach-o 文件及每个 Mach-o 文件所对应的内存起始地址，并根据内存起始地址获取 Mach-o 文件在内

存中的虚拟内存地址；

dylib 文件名称获取函数位于 Mach-o 文件中，Mach-o 是 IOS 系统中可执行文件的格式，对于 Mach-o 文件的结构，具体包括 header structure（头部）、load command（加载命令）和 segment（段），一个 Mach-o 文件可以拥有多个段，每个段可以拥有零个或多个区域（section），每一个段都拥有一段虚拟地址映射到进程的地址空间，同时，一个完整的 Mach-o 文件的末端是链接信息，其中包含了动态加载器用来链接可执行文件或者依赖库所需使用的符号表、字符串等等。dylib 文件名称获取函数位于 Mach-o 文件，且一个 IOS 应用在运行时会加载多个 Mach-o 文件，且每一个 Mach-o 文件都有可能调用到 dylib 文件名称获取函数进行应用内 dylib 文件名称的获取操作，因此为保证判断的准确性，我们需要获取 IOS 系统中所有的 Mach-o 文件，以便于进行后续的操作。

对于系统中 Mach-o 文件的获取，首先可以通过系统函数获取到系统中 Mach-o 文件的个数以及 Mach-o 文件的内存起始地址，具体实现为：

执行 uint32_t c = _dyld_image_count();即通过_dyld_image_count 来获取 IOS 系统中所加载的 Mach-o 文件的个数，然后

执行 for (uint32_t i = 0; i < c; i++)，即通过 for 循环来遍历 Mach-o 文件，接着

执行 onst struct mach_header *header = _dyld_get_image_header(i)，即通过函数_dyld_get_image_header 来获取 Mach-o 文件的内存起始地址，接着

执行 intptr_t slide = _dyld_get_image_vmaddr_slide(i);即通过函数_dyld_get_image_vmaddr_slide 来获取 Mach-o 文件在内存中的虚拟内

存地址。

S102：根据 Mach-o 文件在内存中的虚拟内存地址，解析 Mach-o 文件，得到每个 Mach-o 文件内每个函数的名称和指针；

S103：遍历得到的每个 Mach-o 文件内每个函数的名称，找到每个 Mach-o 文件内的 dylib 文件名称获取函数。

即为了找出 Mach-o 文件中的目标函数，目标函数即为 dylib 文件名称获取函数。具体的：

对 Mach-o 文件进行解析，在 load command（加载命令）内找到 linkedit_segment、symtab_cmd、dysymtab_cmd，通过 linkedit_segment 则可以找到 symtab_cmd 中的 symtab 和 strtab，其中，strtab 内存储着 Mach-o 文件内每个函数的名称，strtab 内存储着 Mach-o 文件内每个函数的索引，而 Mach-o 文件内每个函数的函数指针存储在 section 中，因此需要进一步地从 Mach-o 文件中解析出 section，其过程为：在 Mach-o 文件中获取到对应的 cmd，因为 Mach-o 文件的解析是开源的，故可以按照开源的代码来解析需要查找的 inkedit_segment、symtab_cmd、dysymtab_cmd 等 cmd 地址，故遍历 Mach-o 文件的 Load command 来查找，最终可以查找到 section、symtab、strtab、indirect_symtab。得到每个 Mach-o 文件内每个函数的名称和指针后，便可遍历每个函数的名称，找到每个 Mach-o 文件内的 dylib 文件名称获取函数。

S2：编写一 hook（钩子）函数，所述 hook 函数用于获取 dylib 文件名称获取函数获取的 dylib 文件的名称；即 hook 函数用于将目标函数的内存地址进行替换成设定地址；当目标函数执行时，跳转至设定地址以进行 dylib 文件名称获取函数获取的 dylib 文件名称的获取；目标函数为 dylib 文件名称获取函数。其中，设定地址即为人为编写

的函数地址，用于当 dylib 文件名称获取函数执行时跳转，从而先进行预先编写的功能函数的执行，进而获取到获取 dylib 文件名称获取函数获取的 dylib 文件的名称。

本发明实施例中，hook 函数的原型如下：

HOOK_Function(char* pFuncName, void* pNew, void **pSaveOrg);

该函数中，HOOK_Function 为 hook 函数的名称；

参数 char* pFuncName 标示需要 hook 的函数名称，这里需要 hook 的函数名称为 dylib 文件名称获取函数；

参数 void* pNew 标示替换的函数的内存地址；

参数 void **pSaveOrg 标示存储系统原本的函数的内存地址。

S3：当 IOS 应用调用 dylib 文件名称获取函数进行 IOS 应用内所有 dylib 文件名称的获取时，通过 hook 函数获取 dylib 文件名称获取函数获取的 dylib 文件名称，根据 dylib 文件名称判断是否含有待隐藏 dylib 文件，若是，转到 S4，若否，转到 S5。本发明实施例中待隐藏 dylib 文件即为用于获取 IOS 应用运行状态的 dylib 文件。通过 hook 函数获取 dylib 文件名称获取函数获取的 dylib 文件名称的具体过程为：

1、定义一个函数指针：static const char* (*orig_dyld_get_image_name)(uint32_t image_index); 使其用来存储 dylib 文件名称获取函数_dyld_get_image_name 的内存地址：

2、编写一功能函数 my_dyld_get_image_name，当 _dyld_get_image_name 执行时，通过 hook 函数将目标函数的内存地址进行替换成设定地址，进而执行编写的功能函数 my_dyld_get_image_name，进而获取 dylib 文件名称获取函数获取的 dylib 文件名称。具体的：

调用 hook 函数对_dyld_get_image_name 进行 hook，

具体实现代码为：HOOK_Function("_dyld_get_image_name", my_dyld_get_image_name, (void *)&orig_dyld_get_image_name)

此时，上述的 char* pFuncName 即标示需要 hook 的函数为 _dyld_get_image_name，void* pNew 标示替换的函数为 my_dyld_get_image_name，oid **pSaveOrg 标示将原本的函数 _dyld_get_image_name 地址存储到 orig_dyld_get_image_name 函数指针处。

在 my_dyld_get_image_name 中依次判断当前获取的 dylib 文件名称是否是待隐藏的 dylib 文件的名称，相应代码为：

```
const char* my_dyld_get_image_name(uint32_t image_index)
{
const char* dyName = orig_dyld_get_image_name(image_index);
if (strstr(dyName , "需要隐藏的 dylib 文件名称"))
{
   dyName = orig_dyld_get_image_name(image_index + 1);
}
return dyName;
}
```

如果是需要隐藏的，则调用系统的 orig_dyld_get_image_name 函数得到下一个 dylib 文件名称，并进行判断，通过此方法来对 dylib 文件名称获取函数获取的所有 dylib 文件名称进行判断。

S4：隐藏待隐藏 dylib 文件名称，并将隐藏了待隐藏 dylib 文件名称后的 dylib 文件名称数据返回给 IOS 应用，即返回给 IOS 应用的所有 dylib 文件名称中是不含有待隐藏 dylib 文件的名称；

S5：将 dylib 文件名称获取函数取的 dylib 文件名称返回给 IOS 应用。

通过编写的 hook 函数，获取 dylib 文件名称获取函数获取的 dylib 文件的名称，并根据名称判断是否含有待隐藏 dylib 文件，如果含有，则隐藏，即待隐藏 dylib 文件的名称是不会返回给 IOS 应用的。

在一种实施方式中，为进一步提高待隐藏 dylib 文件隐藏的准确性，避免被 IOS 应用所获取，IOS 应用还用于调用 Mach-o 文件内的 dylib 文件个数获取函数获取 IOS 应用内的 dylib 文件个数；当 dylib 文件名称获取函数获取的 dylib 文件名称中含有待隐藏 dylib 文件名称，则返回给 IOS 应用的 dylib 文件个数为获取的 dylib 文件总个数减去待隐藏 dylib 文件的个数；当 dylib 文件名称获取函数获取的 dylib 文件名称中不含有待隐藏 dylib 文件名称，则返回给 IOS 应用的 dylib 文件个数即为获取的 dylib 文件总个数。具体通过 hook 方式对返回给 IOS 应用的 dylib 文件的个数进行修改。

dylib 文件个数获取函数为_dyld_image_count，该函数为 IOS 的系统函数，通过该函数可以获取 IOS 应用中加载的 dylib 文件的个数，该函数的原型为：

uint32_t _dyld_image_count(void);该函数不需要参数，调用时直接返回 IOS 应用中的 dylib 文件个数，dylib 文件个数获取函数也位于 Mach-o 文件中，其获取方式与 dylib 文件名称获取函数相同。

对于 dylib 文件个数获取函数获取的 Mach-o 文件个数的修改过程，具体为：

定义一个函数指针：static uint32_t (*orig_dyld_image_count)();用来存储_dyld_image_count 函数的地址；同样的编写一功能函数 my_dyld_image_count()，当_dyld_image_count 执行时，通过 hook 方式将目标函数的内存地址进行替换成设定地址，进而执行编写的功能函数 my_dyld_image_count()，进而获取 dylib 文件个数获取函数获取

的 dylib 文件个数，当 dylib 文件名称获取函数获取的 dylib 文件名称中含有待隐藏 dylib 文件名称，则在 my_dyld_image_count()中修改 dylib 文件的个数，该过程相应的代码为：

```
uint32_t my_dyld_image_count()
{
    uint32_t nCount = orig_dyld_image_count();
    --nCount;
    return nCount;
}
```

若待隐藏 dylib 文件的个数为 1，则返回给 IOS 应用的 dylib 文件个数即为获取的 dylib 文件总个数减 1。

对于 hook 的方式，与 dylib 文件名称获取函数被 hook 的方式类似，此时相应 hook 函数为

HOOK_Function("_dyld_image_count", my_dyld_image_count, (void *)&orig_dyld_image_count);

此处该 hook 函数中，char* pFuncName 即标示需要 hook 的函数为 _dyld_image_count，void* pNew 标示替换的函数为 my_dyld_image_count，oid **pSaveOrg 标示将原本的函数 _dyld_image_count 地址存储到 orig_dyld_image_count 函数指针处。

本发明的 IOS 平台隐藏 dylib 文件的方法的原理在于：针对 IOS 应用中当进行应用内 dylib 文件名称获取时必须调用 dylib 文件名称获取函数的特性，通过 hook 的方式，当 IOS 应用调用 dylib 文件名称获取函数进行 IOS 应用内 dylib 文件名称的获取时，通过 hook 函数获取 dylib 文件名称获取函数获取的 dylib 文件名称，并根据名称判断是否含有待隐藏的 dylib 文件，若含有，则隐藏待隐藏 dylib 文件的名称，使最终返回给应用的 dylib 文件的名称中不含有待隐藏 dylib 文件

的名称，即相当于隐藏了应用内的用于获取IOS应用运行状态的dylib文件，避免IOS应用对该类dylib文件的禁止运行操作，保证开发人员能够正常获取到IOS应用运行状态的数据，保证IOS应用调试工作的正常进行。

另外，对应上述IOS平台隐藏dylib文件的方法，本发明还提供一种存储介质，存储介质上存储有计算机程序，计算机程序被处理器执行时实现上述各实施例所述的IOS平台隐藏dylib文件的方法的步骤。需要说明的是，所述存储介质包括U盘、移动硬盘、ROM（Read-Only Memory，只读存储器）、RAM（Random Access Memory，随机存取存储器）、磁碟或者光盘等各种可以存储程序代码的介质。

参见图2所示，对应上述IOS平台隐藏dylib文件的方法，本发明还提供一种电子设备，包括存储器和处理器，存储器上储存有在处理器上运行的计算机程序，处理器执行计算机程序时实现上述各实施例的IOS平台隐藏dylib文件的方法。

本发明实施例还提供一种基于上述IOS平台隐藏dylib文件的方法的IOS平台隐藏dylib文件的系统，包括获取模块、编写模块和判断与执行模块。

获取模块用于获取IOS应用内的所有Mach-o文件及每个Mach-o文件内的dylib文件名称获取函数；编写模块用于编写一hook函数，所述hook函数用于获取dylib文件名称获取函数获取的dylib文件的名称；判断与执行模块用于当IOS应用调用dylib文件名称获取函数进行IOS应用内dylib文件名称的获取时，通过hook函数获取dylib文件名称获取函数获取的dylib文件名称，根据dylib文件名称判断是否含有待隐藏dylib文件，若是，则隐藏待隐藏dylib文件名称，并将隐藏了待隐藏dylib文件名称后的dylib文件名称数据返回给IOS应

用，若否，则将 dylib 文件名称获取函数取的 dylib 文件名称返回给 IOS 应用。

本发明实施例的 IOS 平台隐藏 dylib 文件的系统的原理在于：针对 IOS 应用中当进行应用内 dylib 文件名称获取时必须调用 dylib 文件名称获取函数的特性，通过编写模块，编写一 hook 函数，当 IOS 应用调用 dylib 文件名称获取函数进行 IOS 应用内 dylib 文件名称的获取时，通过 hook 函数获取 dylib 文件名称获取函数获取的 dylib 文件名称，通过判断与执行模块，根据名称判断是否含有待隐藏的 dylib 文件，若含有，则隐藏待隐藏 dylib 文件的名称，使最终返回给应用的 dylib 文件的名称中不含有待隐藏 dylib 文件的名称，即相当于隐藏了应用内的用于获取 IOS 应用运行状态的 dylib 文件，避免 IOS 应用对该类 dylib 文件的禁止运行操作，保证开发人员能够正常获取到 IOS 应用运行状态的数据，保证 IOS 应用调试工作的正常进行。

本发明不局限于上述实施方式，对于本技术领域的普通技术人员来说，在不脱离本发明原理的前提下，还可以做出若干改进和润饰，这些改进和润饰也视为本发明的保护范围之内。本说明书中未作详细描述的内容属于本领域专业技术人员公知的现有技术。

2017108056704

# 说 明 书 附 图

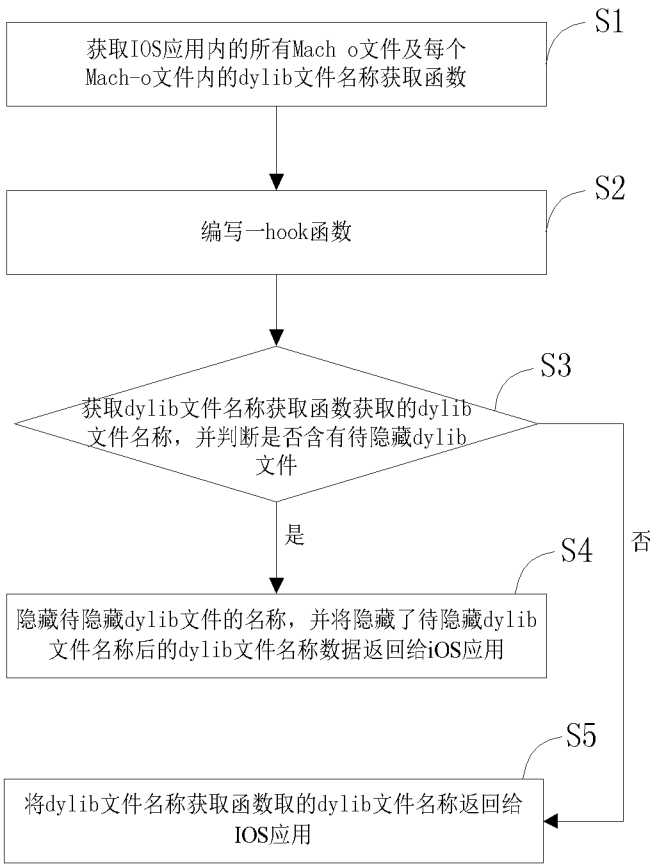


图 1

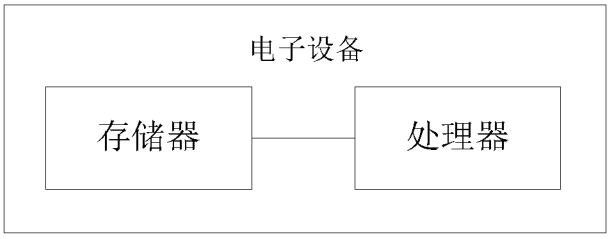


图 2