|  |  |  |  |
| --- | --- | --- | --- |
| 一种基于ptrace实现的动态控制运行中进程能力的解决方案 | | | |
| ■ 文档编号 |  | ■ 密级 | 商业秘密 |
| ■ 版本编号 | Ver 1.0 | ■ 日期 | 2021/02/23 |

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| ■ 版本变更记录 | | | |
| 时间 | 版本 | 说明 | 修改人 |
| 2021/2/23 | Ver1.0 | 新建 | 陈云雄 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| ■ 适用性声明 |
| 本模板用于撰写发明的技术交底书。 |
|  | |

|  |  |  |
| --- | --- | --- |
| 技术交底书  （以下由公司法律顾问填写）  文档编号：  日期： | 申请人： 陈云雄 | 商业秘密 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 交底书名称 | 一种基于ptrace实现的动态控制运行中进程能力的解决方案 | | | |
| 发明人 | 陈云雄 | | | |
|  |  |  | （以下由公司法律顾问填写） | |
| 交底书撰写人 | 陈云雄 |  | 专利类型 | 发明专利 |
| 联系电话 | 15968853154 | 专利负责人 |  |
|  |  | 电话 |  |
| E－mail | chenyunxiong@uniontech.com | E－mail |  |

1. 缩略语和关键术语定义

capabilities：将root用户的权限细分为不同的领域，可以分别启用或禁用。从而，在实际进行特权操作时，如果euid不是root，便会检查是否具有该特权操作所对应的capabilities，并以此为依据，决定是否可以执行特权操作。

文件能力：Permitted:该能力当可执行文件执行时自动附加到进程中，忽略Inhertiable capability。Inheritable:它与进程的Inheritable集合做与操作，决定执行execve后新进程的Permitted集合。Effective: 文件的Effective不是一个集合，而是一个单独的位，用来决定进程成的Effective集合。

进程能力：Permitted: 它是effective capabilities和Inheritable capability的超集。如果一个进程在Permitted集合中丢失一个能力，它无论如何不能再次获取该能力(除非特权用户再次赋予它)。Inheritable: 它是表明该进程可以通过execve继承给新进程的能力。Effecitive: Linux内核真正检查的能力集。

ptrace：ptrace系统调从名字上看是用于进程跟踪的，它提供了父进程可以观察和控制其子进程执行的能力，并允许父进程检查和替换子进程的内核镜像(包括寄存器)的值。其基本原理是: 当使用了ptrace跟踪后，所有发送给被跟踪的子进程的信号(除了SIGKILL)，都会被转发给父进程，而子进程则会被阻塞，这时子进程的状态就会被系统标注为TASK\_TRACED。而父进程收到信号后，就可以对停止下来的子进程进行检查和修改，然后让子进程继续运行。

1. 相关技术背景（背景技术），与本发明最相近似的现有实现方案（现有技术）

传统的unix权限模型进程分为两类：能通过所有权限检测的有效用户ID为０(超级用户)的进程和其他所有需要根据用户和组ID进行权限检测的进程。普通用户需要root权限的某些功能，通常通过setuid系统调用实现。但普通用户并不需要root的所有权限，可能仅仅需要修改系统时间的权限而已。这种粗放的权限管理方式势必会带来一定的安全隐患。需要更细的粒度控制进程权限。

２.１、与本发明相关的现有技术

2.1.1、现有技术一的技术方案

传统的进程能力限制，通常是通过特权级进程设置可执行文件的文件能力，将文件能力的相对应capabilities设置为Permitted和Effective。这样，当可执行文件运行时，即可拥有设置的Effective能力。

2.1.2、现有技术二的技术方案

将可执行文件能力对应capabilities设置为Permitted，不设置Effective。执行可执行文件时，进程的Permitted被设置成enabled，进程的Effective为空。此时可以主动提权，提权范围为当前进程的Permitted。

３、本发明技术方案的详细阐述

3.1、本发明所要解决的技术问题

根据linux的manpage文档可知，内核版本低于2.6.24时，当进程拥有CAP\_SETPCAP能力时，允许当前进程将本进程的Permitted能力赋予其他进程。但当内核版本高于2.6.24时，内核开始支持linux文件能力。此时CAP\_SETPCAP被赋予了不同的含义。此时，拥有CAP\_SETPCAP能将调用者的边界能力集添加进Inheritable能力集。

运行中进程的Effective能力只能期望进程自身或者init进程进行控制。这显然不符合我们期望的使用场景。

这意味着特权级进程丧失了灵活控制其他进程Effective能力的能力。在实际的运用中，我们可能希望动态控制进程的Effective能力，以防止进程在我们不期望的时候，进行了特权级操作。

3.2、本发明提供的完整技术方案

3.2.1、方案简述

本发明提供一整套的动态控制运行中进程Effective和其他能力的解决方案。在本方案实现过程中通过期望控制的运行进程名取得进程ID，通过ptrace调用PTRACE\_TRACEME启动期望控制能力的进程或者PTRACE\_ATTACH控制正在运行中程序的进程，将目标进程变为当前进程的子进程。编写控制当前进程的能力的方法，编写成动态库，加载到进程虚拟内存空间。通过ptrace的向子进程传入参数，并在子进程中调用动态库中的函数进行，参数中限制的能力进行控制。实现动态控制进程能力的作用。

3.2.2、模块介绍

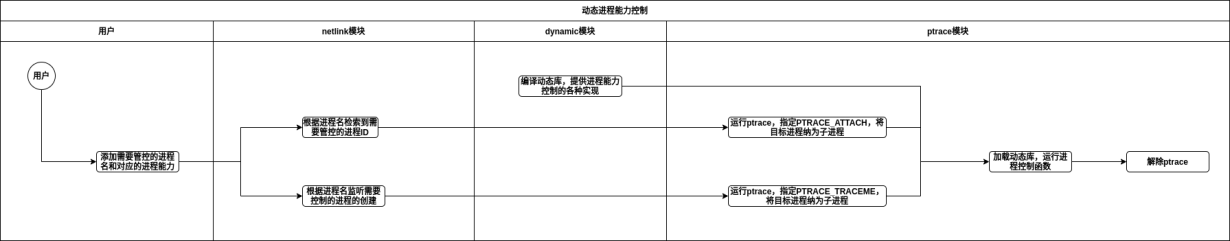
NetLink模块：Netlink套接字家族（英语：Netlink socket family）是一组Linux核心接口（Linux kernel interfaces），可用于进程间通信，Linux内核与用户空间的进程间、用户进程间的通讯。采用NetLink与内核交互，实时监听进程的创建和销毁。当监听到进程创建时，获取当前进程的进程名，并查找被规范的权限范围，实现即时控制的作用。也可以在后续控制中，根据进程名，找到需要控制的进程ID，实现实时控制的作用。

Ptrace模块：作为父进程可以观察和控制其子进程执行。获取到进程路径或者进程ID，将目标进程作为子进程纳入管控。实现控制子进程的各种操作，在本专利方案中，主要负责加载控制能力动态库。并让子进程查询和更新进程能力。

Dynamic模块：提供查询和控制进程能力的各种实现，编译成动态库，方便ptrace进行调用。

3.2.3、具体实现

总体流程如下图所示：



当用户设置了添加控制进程能力的进程名和对应的能力后，netlink模块维护了一个进程名和能力集组成的链表。首先netlink模块先检索当前所有运行的进程中，是否存在对应的进程，如果存在则进行到ptrace模块。如果不存在，则通过netlink与内核的通信，监听进程的产生，如被监听到的进程名为需要控制能力的进程，则进入ptrace模块。

Dynamic模块提供了获取当前进程的方法，动态库libcap.so已经实现了所涉及的方法，在子进程中，通过ptrace调用cap\_get\_proc，并将cap\_t通过ptrace从寄存器返回。并在父进程中，调用cap\_get\_flag检索进程当前的Permitted和Effective。在父进程通过cap\_set\_flag，设置好更新的cap\_t后，通过ptrace传入子进程，在子进程调用cap\_set\_proc和cap\_clear对进程能力进行控制。

Ptrace模块中，应该加载shared动态库，加载的方式，利用dlopen为通过计算得出malloc，free和\_\_libc\_dlopen\_mode在libc中的真实地址。实现代码加载器函数injectSharedLibrary，设置加载器底部INTEL\_INT3\_INSTRUCTION，作用为执行代码后，将控制权返回给控制器，恢复内存空间。将加载器代码通过ptrace\_write注入目标进程虚拟内存空间。执行ptrace\_cont，回到子进程，利用ptrace使用malloc，\_\_libc\_dlpoen\_mode和free完成动态库加载。函数地址在之前已经获取到了，只要将路径通过ptrace\_getregs和ptrace\_write写入子进程即可。

而后，同样在dlopen获取cap\_get\_proc，cap\_set\_proc和cap\_set\_flag地址。先通过cap\_get\_proc获取cap\_t，通过ptrace返回给父进程。在父进程中，查看进程的Permitted能力，如果当前要求子进程Effective能力提升，则最大不应该超过Permitted。可以删除Effective和Permitted能力。设置好cap\_t后，通过ptrace\_write回写入子进程虚拟内存空间。ptrace\_cont将控制权交回给子进程。执行控制能力函数cap\_set\_proc。

3.3、本发明技术方案带来的有益效果

1. 动态添加运行中进程的能力。当某些进程开始运行时，又不方便停下来，在某些时刻，需要某些特殊的进程能力来执行一定操作时，此方案提供了很好的解决方案。

2. 动态移除运行中进程的能力。当某些进行某些时候拥有特殊进程能力，用户又不期望有这种能力时，可以动态移除进程的这些能力。

3. 提供了一个集中到进程来控制目标进程的能力，方便管理进程能力。

4、本发明的技术关键点和欲保护点是什么

本发明中的关键点：

1. netlink监听。通过netlink与内核通信，获取创建和注销的进程信息。

2. ptrace注入。通过ptrace注入的方式，动态加载动态库。

3. ptrace对进程能力控制。通过ptrace调用动态库中的方法，控制指定进程的能力。

本发明的保护点：

1. netlink和ptrace组成的动态控制运行中进程能力的方案。

4、附件：

参考文献（如专利/论文/标准等）

1. <https://man7.org/linux/man-pages/man7/capabilities.7.html>
2. https://man7.org/linux/man-pages/man2/ptrace.2.html

3. https://github.com/gaffe23/linux-inject

4.https://driverxdw.github.io/2020/07/06/Linux-ptrace-so%E5%BA%93%E6%B3%A8%E5%85%A5%E5%88%86%E6%9E%90/

5. <<Linux UNIX系统编程手册 下>>