# CVNWARE平台权限管理方案

## 背景

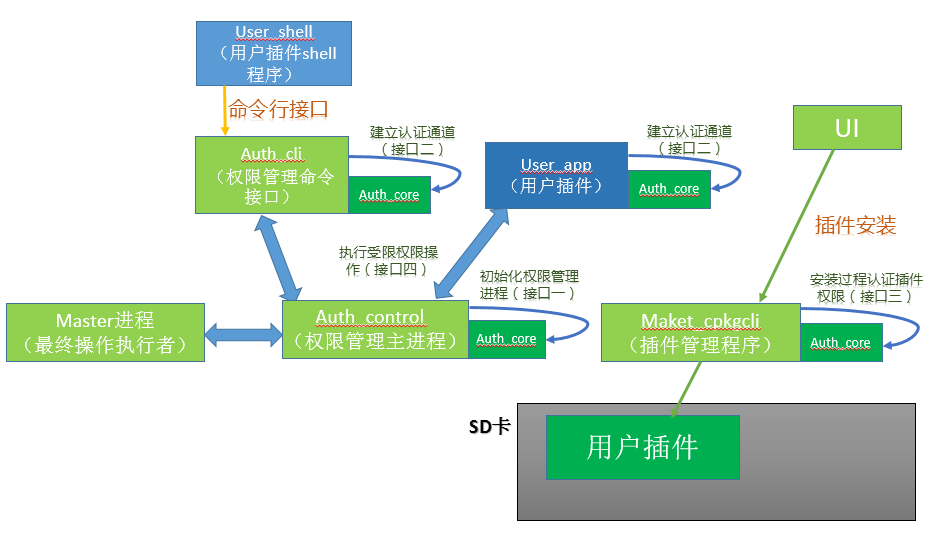
用户插件权限管理，类似于手机上安装app的时候，弹出此应用需要获取的权限操作类型，通过用户的许可后，才可完成安装流程。因此CVNWARE平台权限管理的目标如下：

* 约束用户插件行为，维护产品稳定。
* 易于管理插件和审核。
* 关键操作流上，可形成可追溯的日志系统。

总而言之，目的在于提高产品稳定性、更易管理与维护。

## 权限管理功能点

**功能框图**



**名词解释：**

设备权限表-设备开发出来的权限操作集合

用户插件权限表-用户插件申请的权限操作集合

**Auth\_core**：权限管理核心程序，以lib库的形式存在，被其他进程加载。主要负责的功能有：

* 维护设备权限表。
* 维护用户插件权限表。
* 通过IPC消息，建立用户插件与auth\_control管理程序认证通道。

**Auth\_control**：权限管理主进程，主要功能有

* 分析配置文件，获取管理权限操作，通过auth\_core接口设置到设备权限表中。
* 解析用户插件权限操作IPC消息，执行操作。

**Auth\_cli：**提供命令行接口供shell程序调用。

**Market\_cpkgcli：**安装插件的过程中，从云端获取用户插件权限，通过auth\_core接口设置到用户插件权限表中。

**Master：**cvnware平台系统管理进程，当用户权限操作被许可后，auth\_control通知master进程执行相关操作。

#### Auth\_control主进程初始化

* 1. **方案一：控制进程初始化过程（推荐）【接口一】**

1. 控制进程使用单例模式，整个系统只可启动一个。
2. 分析配置文件，获取管理权限操作，通过auth\_core接口添加到设备权限表中。
3. 创建认证通道，返回服务端socket，用于接收用户插件权限操作ipc消息。
4. 打开master进程ipc通道，用于向maser发送操作消息。
   1. **方案二：由master进程加载权限管理lib**
5. master进程启动时，加载权限管理模块。
6. 从master配置中获取权限列表，分析配置文件，获取管理权限操作，通过auth\_core接口添加到设备权限表中。
7. 创建认证通道服务端，用于接收用户插件权限操作ipc消息，执行相应操作。
   1. **配置文件，适用于1.1的方案，权限列表格式（重点）**

*[操作模块]*

*操作命名1：操作命令字2*

*操作命名2：操作命令字2*

*例如：*

*[udhcp]*

*Udhcp.start : ipc.Udhcp.start*

*Udhcp.stop : ipc.Udhcp.stop*

*[iptables]*

*iptables.nat.delete : ipc.iptables.nat.delete*

*iptables.nat.add : ipc.iptables.nat.add*

操作模块-用户申请权限，以操作模块为单位。一个操作模块，可细分成过个操作。

操作命名-操作命名可用于shell调用接口，命名格式如：操作模块.xxxx。

操作命令字-操作命名字可用于ipc消息命令字。

* 1. **权限列表**
* Udhcp拨号功能权限
* Pppoe拨号功能权限
* 有线接口功能权限
* 无线接口功能权限
* 防火墙功能权限
* DNS功能权限
* NTP功能权限
* 重启功能权限

#### 用户插件建立认证通道

* 1. **建立过程【接口二】**

1. 通过auth\_core打开认证通道，返回认证通道客户端socket（对应着1.1 3）。

#### 认证用户插件权限

* 1. **方案一：文件签名（当前推荐）**

1. 云服务器生成一对RSA密钥，将公钥存储到路由中。
2. 云服务器使用密钥对App权限文件进行文件签名。
3. 路由器从云服务器下载App权限文件与文件签名。
4. 路由器对App权限文件进行签名认证。
5. 签名认证成功后，解析权限文件。

缺点：

1、没有隐私，公钥是公开，谁都可以来解密，看到加密内容。但没有私钥是无法篡改的。

2、密钥非动态生成，依靠个人来维护，容易被盗取，复制。

* 1. **方案二（数字证书）**

1. 云服务器发布数字证书，存储到路由中。
2. 路由使用数字证书与云服务器建立https通道。
3. 云服务器动态生成一对RSA密钥，使用密钥加密权限文件，将公钥通过https通道发给路由。
4. 路由使用动态生成公钥解密权限文件。
5. 解析权限文件。

讨论：

此方案有两种方式，一是自建签名中心，发布数字证书，二是使用官方CA机构发布数字签名。

从未来的角度来看，官方CA机构发布的数字签名是必须的，这是最为安全的方式来构建与服务器的安全通道。

* 1. **认证流程**

1. 从网页上下载插件，进行插件安装。
2. Sd卡安装成功后，进行权限认证。

* 如果安装在sd卡中失败，退出整个安装流程，返回失败给网页。

1. 选定认证方案，进行插件的权限获取。

* 获取失败，退出整个安装流程，返回失败给网页。

1. 解析权限文件，通过auth\_core接口添加到用户插件权限表中，同时也存储到Appname.profile文件中。
   1. **权限文件格式**

使用xml格式来存储用户插件权限文件。

<authlist>

<module name ="udhcp" > </module>

<module name ="pppoe" > </module>

<module name ="iptables" ></module>

</authlist >

#### 执行权限操作

权限操作调用过程不可包含长时间阻塞操作，用户需要等待执行结果。

* 1. **权限操作流程【接口四】**

1. 使用认证通道客户端socket，发送操作消息至control程序。
2. Control程序从消息中提取操作命令字，通过auth\_core接口检测操作是否具有权限

* 如果不具有权限，返回权限受限标志。
* 如果具有权限，则继续。

1. 操作命令带参数的，需要检测参数合法性（合法性参考4.3）。
2. 返回操作结果给用户
   1. **通过命令使用权限操作**
3. 用户shell可调用auth\_cli –c 操作命令名 –p 参数[可选]进行权限操作。
4. auth\_cli使用认证通道客户端socket，发送操作消息至control程序。
5. Control程序从消息中提取操作命令字，通过auth\_core接口检测操作是否具有权限

* 如果不具有权限，返回权限受限标志。
* 如果具有权限，则继续。

1. 操作命令带参数的，需要检测参数合法性（合法性参考4.3）。
2. 返回操作结果给用户
   1. **IPC消息格式。**

command=操作命令字；param=命令参数；

命令参数需做检测，只可为数字，字母与空格，不可包含特殊字符，如； 。