# TP-Link SR20

****简述：****3月26号TP-Link Smart Home Router (SR20) 的远程代码执行漏洞，此远程代码执行漏洞允许用户在设备上以 root 权限执行任意命令，该漏洞存在于 TP-Link 设备调试协议(TP-Link Device Debug Protocol 英文简称 TDDP) 中，TDDP 是 TP-Link 申请了[专利](https://patents.google.com/patent/CN102096654A/en)的调试协议，基于 UDP 运行在 1040 端口



TP-Link SR20 设备运行了 V1 版本的 TDDP 协议，V1 版本无需认证，只需往 SR20 设备的 UDP 1040 端口发送数据，且数据的第二字节为 0x31 时，SR20 设备会连接发送该请求设备的 TFTP 服务下载相应的文件并使用 LUA 解释器以 root 权限来执行，这就导致存在远程代码执行漏洞

**环境**：

Qemu 2.4.0：github下载2.4.0版本的qemu，编译安装时候

./configure –target-list = arm-soft ,mips-soft ,mipsel-soft 等，这样编译时间会大大缩短。

IDA6.8（一些ida脚本，揭秘路由器0day书上有）

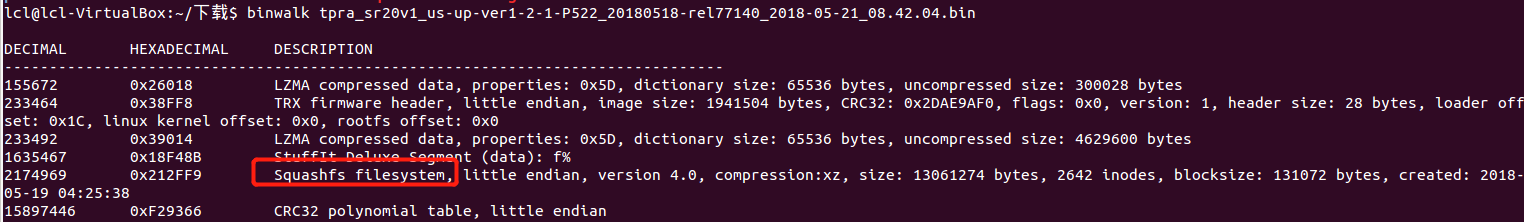
Binwalk（github上安装，不用apt安装，不然会少一些功能，安装的看揭秘路由器0day书上有）

**复现过程：**

使用 binwalk 查看该固件

（[https://www.tp-link.com/us/support/download/sr20/#Firmware](https://www.tp-link.com/us/support/download/sr20/" \l "Firmware)

从 TP-Link SR20 设备[官网](https://www.tp-link.com/us/support/download/sr20/" \l "Firmware)下载固件）



使用 binwalk 把 Squashfs filesystem 从固件中提取出来

binwalk -Me + 固件，binwalk 会在当前目录的 \_+bin文件名 目录下生成提取出来的固件里的所有内容，进入到该目录





squashfs-root 目录就是我们需要的固件文件系统

在该文件系统目录下查找存在漏洞的 tddp 文件并查看文件类型可以看到该文件是一个 ARM 架构的小端(Small-Endian)32 位 ELF 文件





## 搭建 ARM QEMU 虚拟机环境

从 Debian [官网](https://people.debian.org/~aurel32/qemu/armhf/)下载 QEMU 需要的 Debian ARM 系统的三个文件:

<https://people.debian.org/~aurel32/qemu/armhf/>

把以上三个文件放在同一个目录执行以下命令

sudo tunctl -t tap0 -u `whoami` # 为了与 QEMU 虚拟机通信，添加一个虚拟网卡

$ sudo ifconfig tap0 10.10.10.1/24 # 为添加的虚拟网卡配置 IP 地址

$ qemu-system-arm -M vexpress-a9 -kernel vmlinuz-3.2.0-4-vexpress -initrd initrd.img-3.2.0-4-vexpress -drive if=sd,file=debian\_wheezy\_armhf\_standard.qcow2 -append "root=/dev/mmcblk0p2 console=ttyAMA0" -net nic -net tap,ifname=tap0,script=no,downscript=no -nographic

虚拟机启动成功后会提示登陆

用户名和密码都为 root

配置网卡IP：

ifconfig eth0 10.10.10.2/24

此时 QEMU 虚拟机可以与宿主机进行网络通信

现在需要把从固件中提取出的文件系统打包后上传到 QEMU 虚拟机中

压缩固件文件系统目录下的整个文件

tar -cjpf squashfs-root.tar.bz2 squashfs-root/

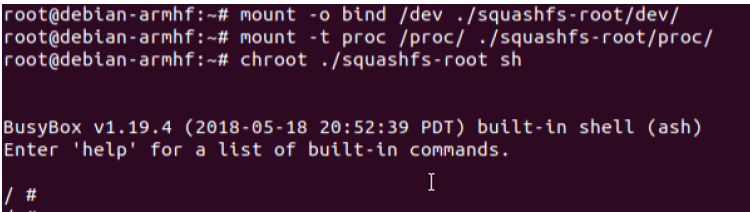
使用 chroot 切换根目录固件文件系统

$ mount -o bind /dev ./squashfs-root/dev/

$ mount -t proc /proc/ ./squashfs-root/proc/

$ chroot squashfs-root sh # 切换根目录后执行新目录结构下的 sh shell

使用 chroot 后，系统读取的是新根下的目录和文件，也就是固件的目录和文件 chroot 默认不会切换 /dev 和 /proc, 因此切换根目录前需要现挂载这两个目录



## 搭建 TFTP Server

在宿主机安装 atftpd 搭建 TFTP 服务

sudo apt install atftpd

* 编辑 /etc/default/atftpd 文件，USE\_INETD=true 改为 USE\_INETD=false
* 修改 /srv/tftp 为 /tftpboot

$ mkdir /tftpboot

$ chmod 777 /tftpboot

$ sudo systemctl start atftpd # 启动 atftpd

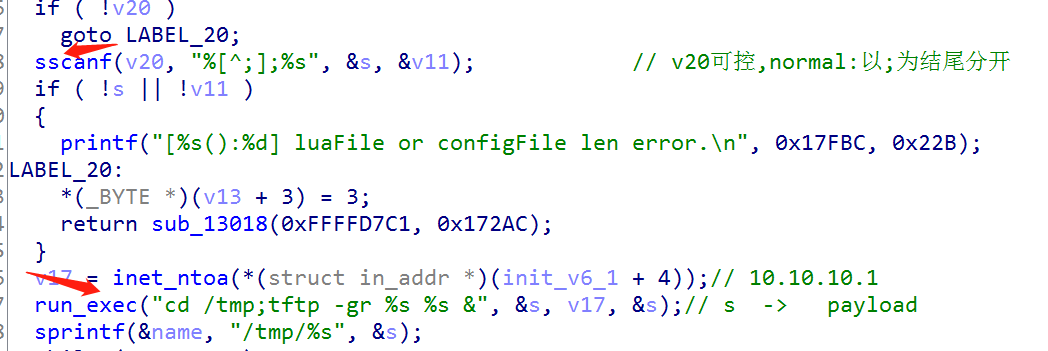
如果执行命令 sudo systemctl status atftpd 查看 atftpd 服务状态时

提示 atftpd: can't bind port :69/udp 无法绑定端口

可以执行 sudo systemctl stop inetutils-inetd.service 停用 inetutils-inetd 服务后

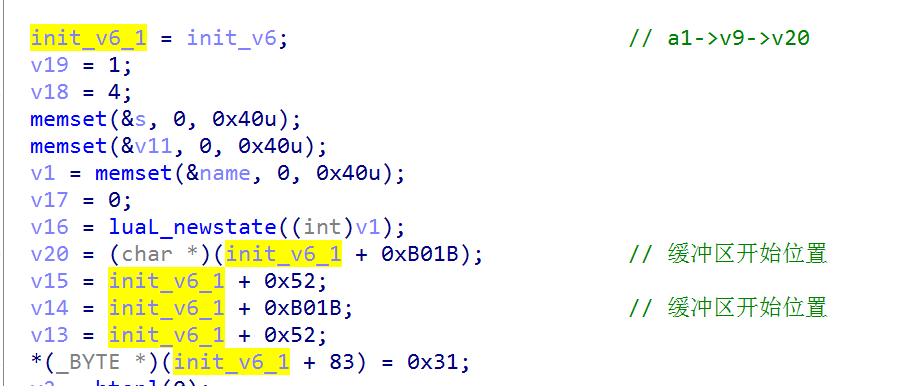
再执行 sudo systemctl restart atftpd 重新启动 atftpd 即可正常运行 atftpd

漏洞细节分析，**最精彩一部分**：



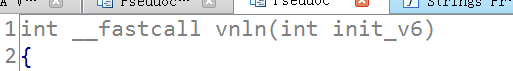
漏洞主要出现在sscanf函数上，且v20是可控的，这样就不需要一定要按sscanf指定的表达式而传递参数，如果v20没有’;’，而是’|’，则s就是可控的，s的传递会到run\_exec函数，tftp -gr + 文件名+ip地址

现在看看谁调用v20,向上向前找

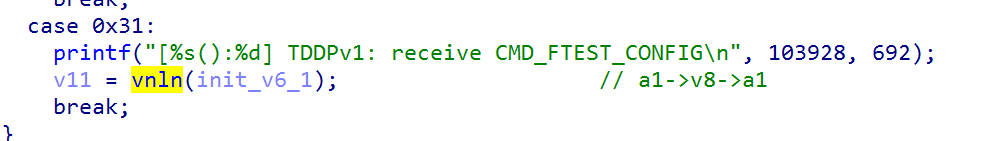


v20 由init\_v6\_1传，由init\_v6 变量名都是自己命名

init\_v6是函数的参数



看看谁调用了vnln函数，按‘x’键



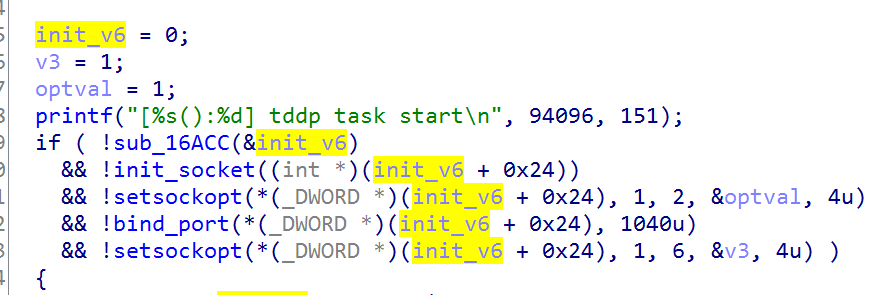
继续往上找，谁调用了init\_v6\_1

...

....

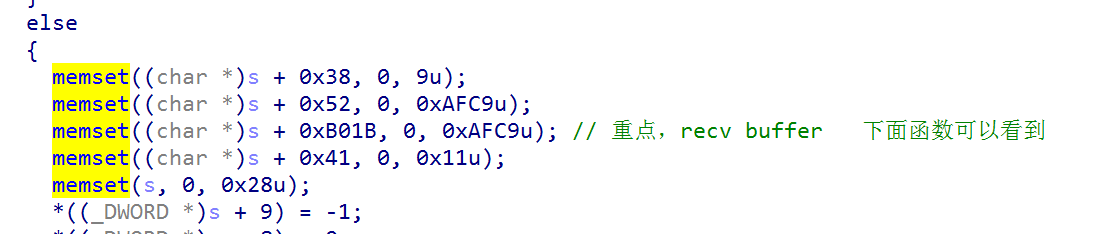
...

到了sub\_936c函数，没有传变量了，



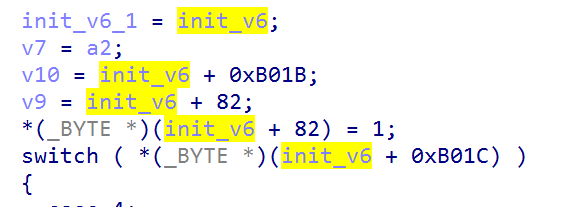
Init\_v6初始化为0，通过下面函数可以判断这是个socket套接字有关的函数，

进入sub16acc函数，

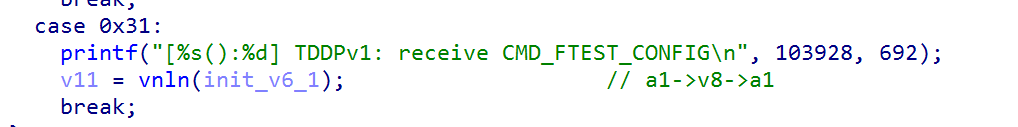


S+0xb018是recv buffer的地方

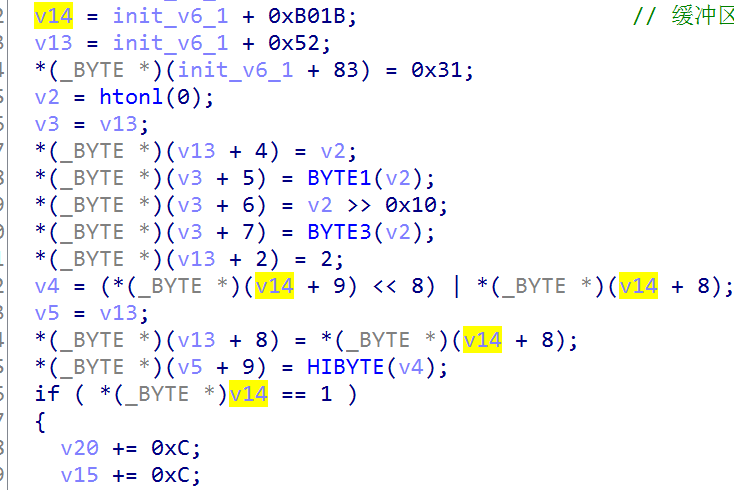
还有一点：



判断init\_v6+0xB01C第一字节是不是0x31，如果是，则进入漏洞函数，这个主要根据上面漏洞点先回溯的找到这个漏洞出发点，然后正向查看如何触发这个漏洞

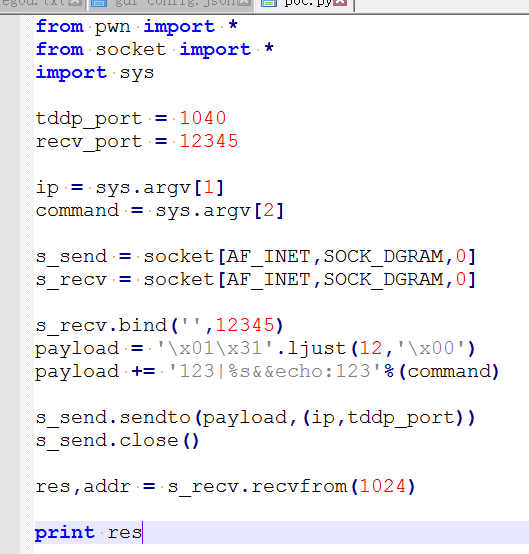


进入这个函数后，会判断是不是缓冲区第一字节+0xc01B是不是等于1



等于1，则加0xC字节，

分析poc:



Tddp是1040端口，UDP协议，

0131+12个字节的垃圾数据+ payload