Zingbox

Watching You through the Eyes of Celia, a Telepresence Robot

Dan Regalado

Robotic telepresence is a next-generation technology that allows a person in one location to replicate himself in another. The remote person can see you, hear you, interact with you, and move all around your location. But wait a second! What if the person behind the robot is not who you think he is? What if the robot gets compromised, and now the attacker is watching you and your surroundings?

In this whitepaper, all the findings learned while security testing a telepresence robot are presented, as well as the countermeasures implemented by the vendor.

Findings Overview

For our research, we picked one of the most widely known telepresence robots on the market: VGo, nicknamed "Celia", from Vecna (see Figure 1). We then performed a vulnerability assessment to try to gain unauthorized access and control of it.

Figure 1: VGo ("Celia") from Vecna Robotics

A total of five vulnerabilities were reported to the manufacturer via ICS-CERT:

- CVE-2018-8858: Insufficiently Protected Credentials – Wi-Fi, XMPP – Patch Pending
- CVE-2018-8860: Cleartext Transmission of Sensitive Information – Firmware - Patched
- CVE-2018-8866: Improper Neutralization of Special Elements – RCE - Patched
- CVE-2018-17931: Improper Access Control (USB) – Patch Pending
- CVE-2018-17933: Improper Authorization (XMPP Client) – Patch Pending

Note: Vecna responded promptly and began fixing the issues we discovered.

Telepresence Robot Overview

The way Celia works is pretty simple. An XMPP chat client is installed on the computer that controls the robot remotely via video calls over the VGoNet Cloud Network. Once connected, the caller's face is displayed on the device's screen, and the caller can start controlling the robot using controls in the client interface (Figure 2).

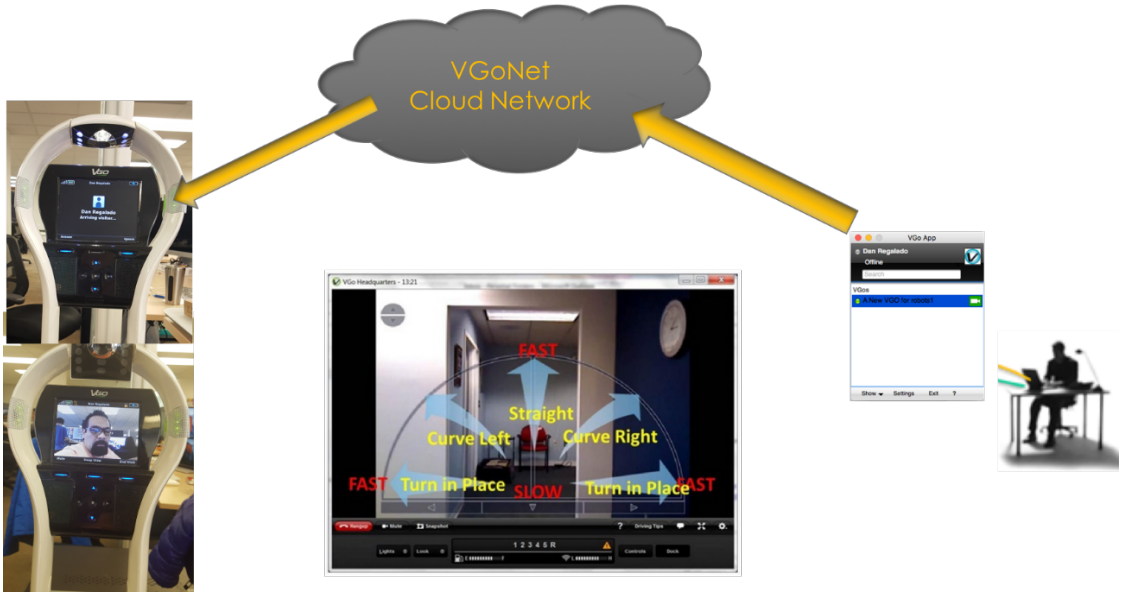

Figure 2: VGo communication flow

A short demo from the manufacturer can be found here:

https://www.youtube.com/watch?v=OPx5XppT2lc

The following are some of the main functions on the VGo robot:

- It can receive voice calls.
- It can send and receive video streaming.
- Using text-to-speech technology, the robot can speak chat messages it receives.
- The robot can move around at different speeds and roam between access points.
- It can take pictures in the remote location.
- It can recognize speech it hears through its environmental microphone.

There is no doubt that robotic telepresence is a great technology from which humans can benefit. Some of these benefits are described in the next section.

Parents can talk to beloved relatives from remote locations in difficult times (Figure 3).

Figure 3: Remote visits

A remote doctor can visit patients at home (Figure 4).

Figure 4: Remote doctor visits

A remote expert can provide support and discuss issues with technicians on site (Figure 5).

Figure 5: Remote technical support

Last but not least, disabled children can attend schools remotely (Figure 6).

Figure 6: Disabled children at school remotely

As the above examples show, the benefits are clear. We only need to make sure the person controlling the robot is the expected one. Otherwise, an attacker could be inside a body shop, a school, a hospital, or even your own home! Unfortunately, it was possible to compromise the robot locally with a USB thumb drive, and remotely through an unsecured web page. Once inside the device, the user's confidentiality could be seriously infringed. In the next sections, all these issues are described in detail.

Intercepting firmware when updating the robot – CVE-2018-8860

Note: The vendor addressed this issue in firmware version 3.0.3.53662.

Because the robot performs firmware updates over HTTP, an attacker with access to the same network segment where the robot is connected can intercept the update. Figure 7 shows the firmware update being captured after sniffing the network.

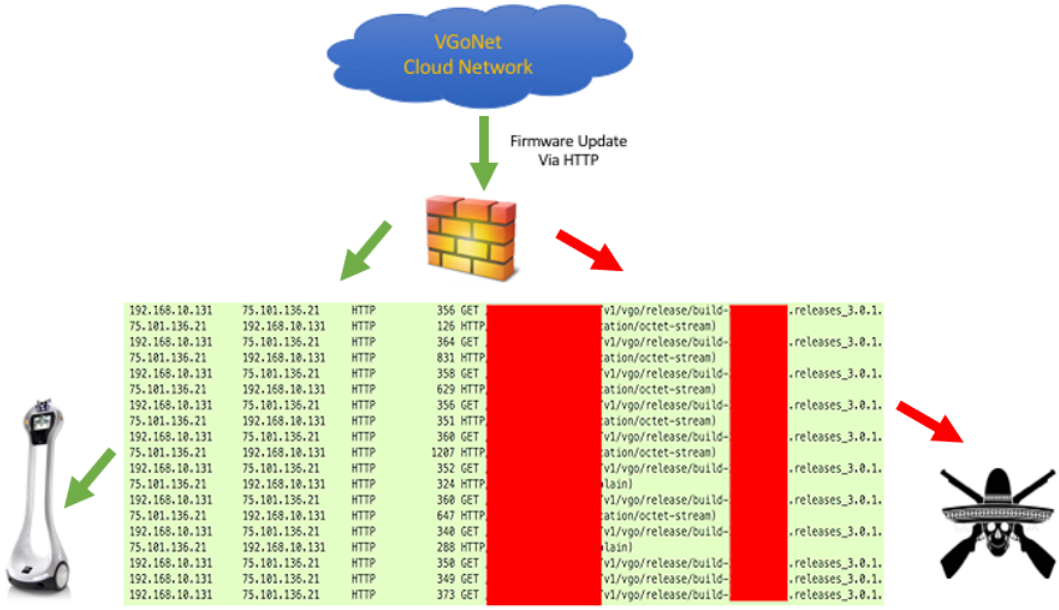

Figure 7: Intercepting a firmware update

Accessing sensitive information inside the firmware – CVE-2018-8858

Once we intercept the firmware, we can use tools like binwalk to extract and mount the content of the file system, if it happens to be a known one. In this case, it's a UBIFS file system, which can be partially handled by the tool. Once mounted, we focused on finding a web interface so that we could compromise the robot remotely if we could find a vulnerability. After digging through the file system, we found a CGI program along with the credentials to access it. The credentials were stored in clear text, as partially shown at Figure 8.

```
danux@XpL0iT:~/_UBIFS_rootfs.img.extracted/ubifs-root$ ls
appfs  bin  BUILDTIME  config  dev  etc  home  include  lib  linuxrc  mnt  opt  proc  root
danux@XpL0iT:~/_UBIFS_rootfs.img.extracted/ubifs-root$ ls var/www/cgi-bin/
[redacted]_cmd  test_cgi
danux@XpL0iT:~/_UBIFS_rootfs.img.extracted/ubifs-root$ cat etc/httpd.conf
A:127.0.0.1
A:192.168.0.0/16
A:10.0.0.0/8
D:*
/cgi-bin:ego:[redacted]
```

Figure 8: Firmware exploration

We connected to the CGI script, which turned out to be a developer's tool that was not supposed to be included in production. It could run limited commands on the robot, probably for diagnostics, such as those to view running processes, view logs, reboot the robot, and see network connections. The interface and a list of the running processes are shown in Figure 9.

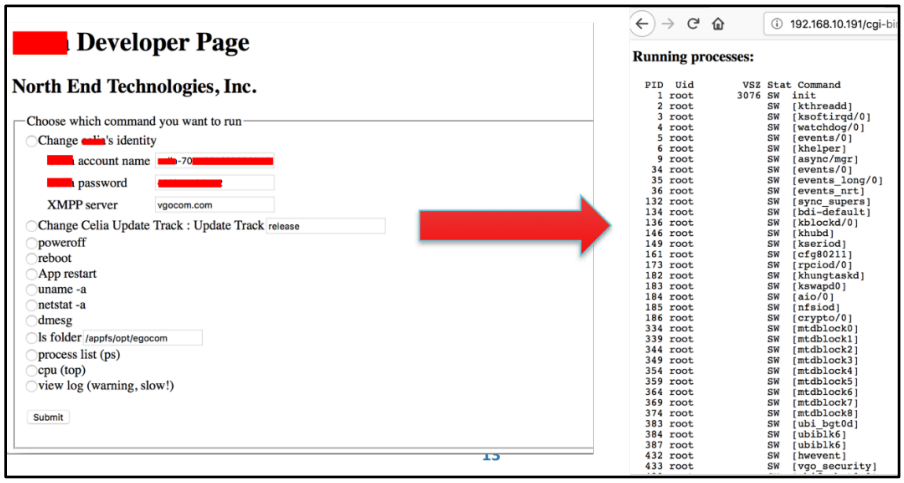

Figure 9: VGo hidden developer's interface

Gaining remote code execution – CVE-2018-8866

Note: The vendor addressed this issue in firmware version 3.0.3.53662.

Unfortunately, most of the `GET` parameters of the CGI are vulnerable to command injection due to the lack of input validation. In Figure 10, the `"celiatrack" GET` parameter is not properly validated and its content is copied into the `/config/startup.script`, which is run every time this option is called. A simple backtick trick will provide arbitrary system execution. An injection would look like this:

```
celiatrack=`uname -a`
```

The vulnerable part of the code is shown in Figure 10.

```
NEWCELIATRACK=`echo "$QUERY_STRING" | sed -n 's/^.*celiatrack=\([^&]*\).*$/\1/p' | sed "s/%20/ /g"| sed "s/%2F/\//g"`

celia_cmd.cgi [+]
        ;;

        changetrack)
      echo "<h3>Changing Celia's update track:</h3>"
      echo "New Celia Update Track is $NEWCELIATRACK <BR>"
          c /config/startup.script /config/startup.old
          echo "export CELIA_TRACK=$NEWCELIATRACK" >> /config/startup.script
          echo "<pre>"
```

Figure 10: Vulnerable parameter found in the CGI script

Another important point to mention is that the CGI script runs with root privileges. Therefore, after we have access to arbitrary shell commands, we can gain unauthorized root access to the robot (Figure 11).

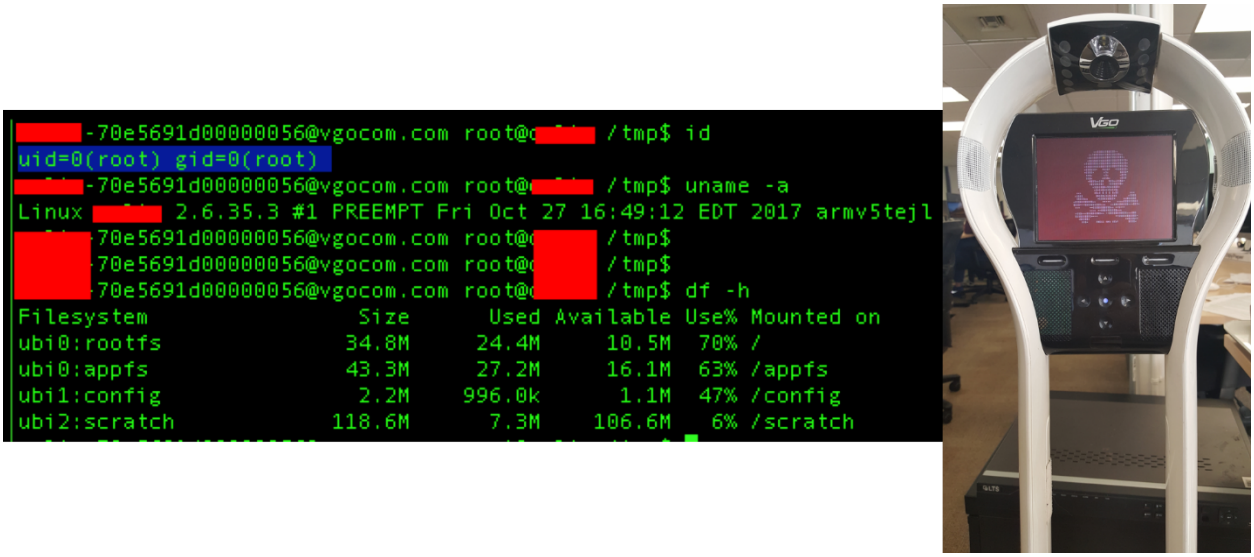

Figure 11: Robot compromised remotely

At this point, an attacker can start attacking other systems located in the same network segment as the robot!

Gaining Code Execution via USB – CVE-2018-17931

We found another way to gain code execution as root inside the robot. This approach requires physical access to the USB slot 2 located in the back of the robot (Figure 12). If a USB containing a file with the name `startup.script` inside a `config` folder in the root partition is plugged in, the content inside that file will be executed as root without restrictions. The robot just needs to be rebooted to trigger the execution. At the time of this writing, the vulnerability was released to the manufacturer but hasn't been fixed yet, so we cannot provide details of an exploitation.

Figure 12: USB Slot 2 used for root access

Robot Technical Details

Once inside the robot, we can learn its technical details:

- CPU (video): i.MX27 (NXP)
- OS: Linux ARMv5 – 32 bits – kernel 2.6.35.3
- Voice: i.MX31 (NXP) – Text to speech
- UbiFS file system
- Gloox 1.0.20 – XAMP (Jabber) client
- QT interface: Robot and XMPP client

Finding other sensitive information inside the robot - CVE-2018-8858

Stealing sensitive credentials

Once inside the robot, Wi-Fi and robot XMPP credentials were found in clear text. This finding combined with the USB attack shown previously allows an attacker to walk inside a hospital, plug the USB into the robot, reboot it, steal hospital Wi-Fi credentials, walk out of the hospital, connect to an access point remotely and start attacking other network assets in the corporation! This scenario is shown in Figure 13.

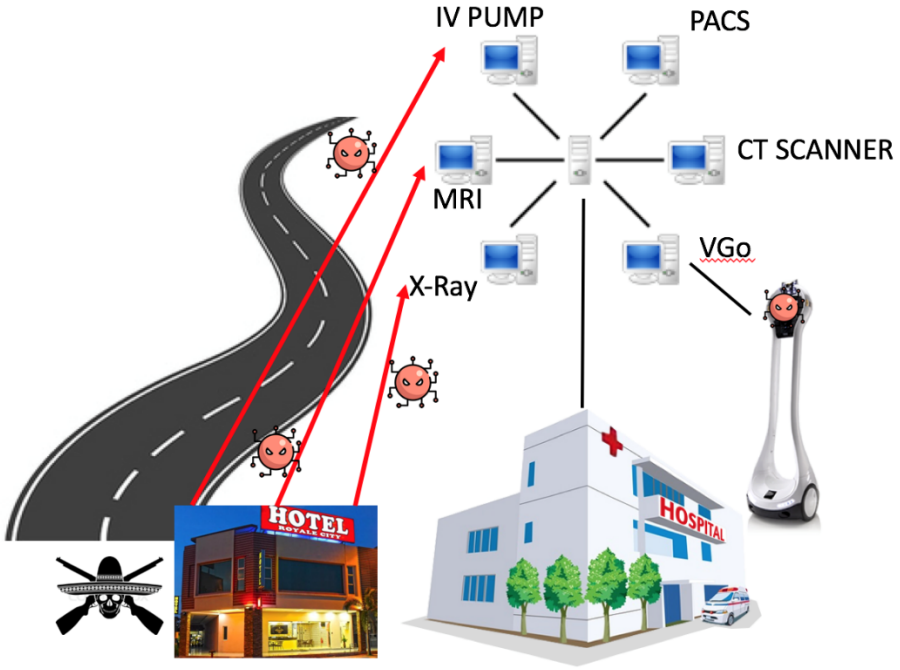

Figure 13: Attacker using stolen Wi-Fi credentials to launch a remote attack

Stealing chat conversations

We also found chat information in log files, allowing us to read and steal text messages sent from doctor to patient or teacher to student. See Figure 14 in which the left side is the legitimate text sent from the VGo XMPP client and the right side shows how an attacker can look at the same content from inside the robot.

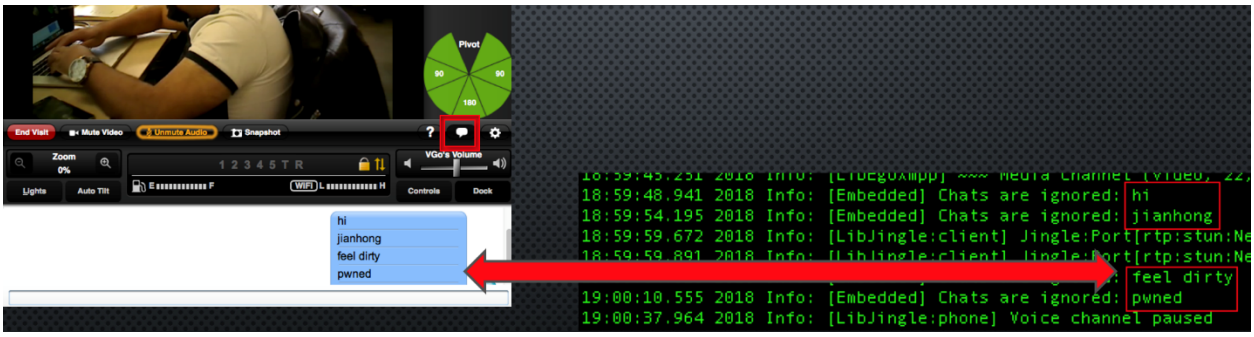

Figure 14: Reading chat conversations

Stealing pictures

The remote user controlling the robot has the option to take pictures of people and surroundings wherever the robot goes. The user takes pictures with the robot's camera and sends them through an encrypted channel to the VGoNet Cloud from which the VGo XMPP client retrieves them. However, the pictures are temporarily stored locally in the robot's file system, and since an attacker is already inside, he can easily steal the pictures as soon as they are created. Every new picture is stored with the same name as `RemoteSnapshot.jpg` in the `/tmp` directory (see Figure 15).

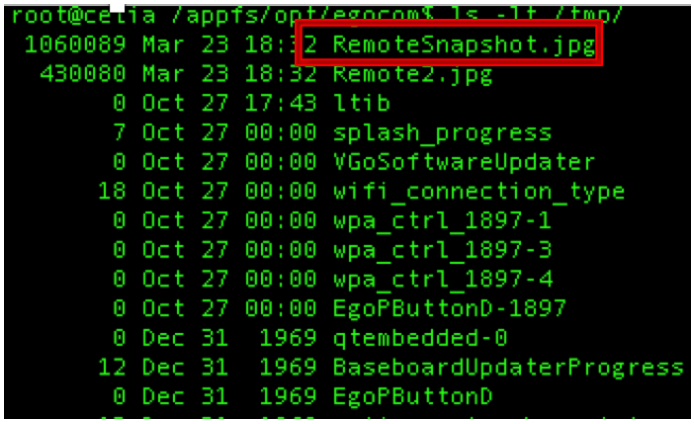

Figure 15: Pictures stored temporarily in the file system

Executing sensitive commands on the robot via XMPP clients – CVE-2018-17933

The robot can receive system commands from an authenticated XMPP client, normally a doctor or teacher. However, due to the sensitivity of those commands, they should only be available for VGo system administrators.

Note: Unfortunately, once inside the robot, these commands can be executed directly on the robot without needing to use an external XMPP client.

The commands are parsed by Celia in the `ReceivedChatText()` class. The address location inside the binary is shown below:

```
.text:004218AC    ego::policy::c::CeliaPolicy::ReceivedChatText()
```

Some of the more sensitive commands available are listed below:

- `Reboot` – Reboot the robot remotely
- `GetLogs` – Transfer sensitive logs related to system and robot activities over the network
- `GetVersion` – Display the version of the robot release
- `GetTop` – Display memory usage information
- `VideoRawDump` – Record video streaming and then send it to the client. This is an ideal feature for an attacker to monitor live video streaming! (This command is potentially the most sensitive of all.)

A quick proof of concept can be seen in Figure 16. The VGo client sends the command `${TIME}` (Request from a doctor or "Dr") and the robot responds with the local hour (Response from the robot).

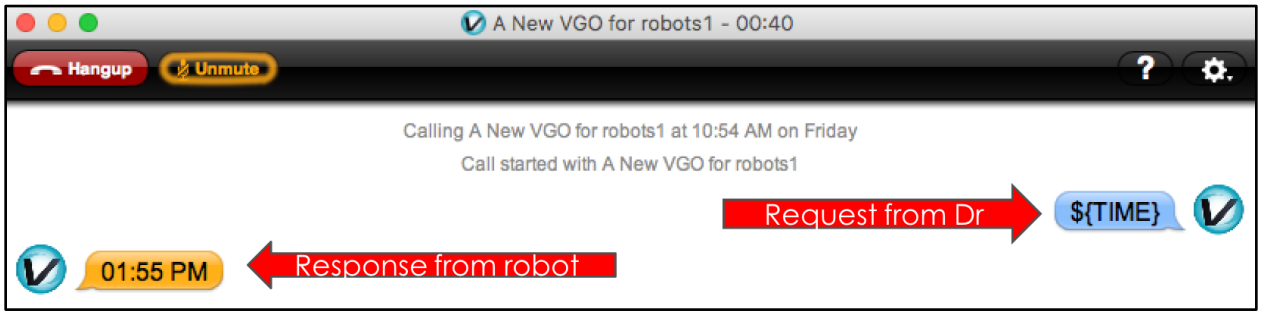

Figure 16: Remote command execution via XMPP client

Monitoring victims via live video streaming

As mentioned above, this might be the most sensitive command since it can affect the confidentiality of the person where the robot is located. An attacker can capture live video streaming remotely and start watching the victims live! This scenario is shown in the Figure 17.

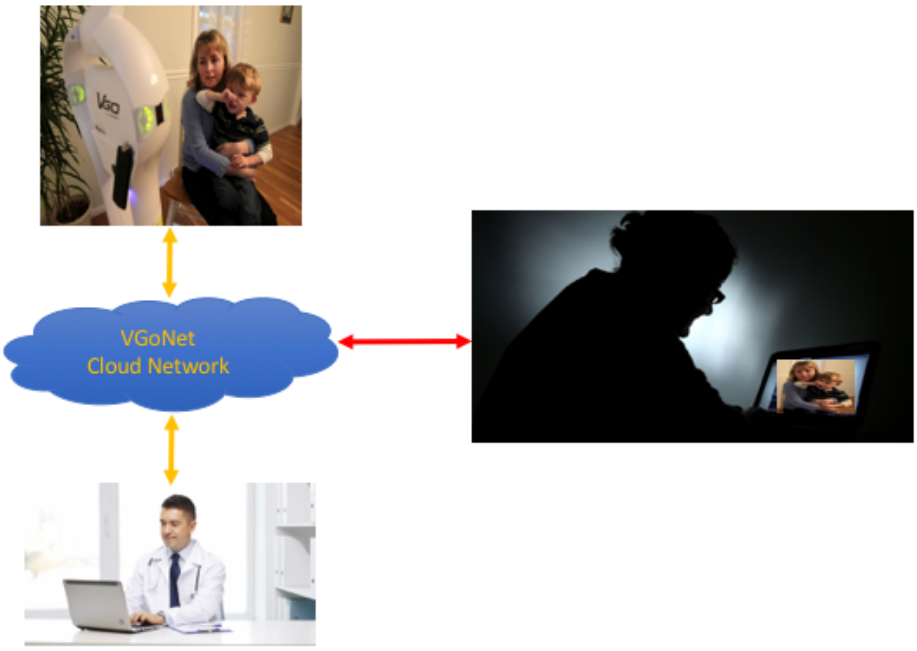

Figure 17: Watching through Celia's eyes

But how does it work behind the scenes? A single command can instruct the robot to start recoding video frames, which are dumped temporarily in H.264 format to the file system. Another command will ask the robot to stop recording. (See "Starting raw dump of H264 frames" and "Stopping raw dump of H264 frames" in Figure 18.) These commands can be issued directly by an attacker inside the robot.

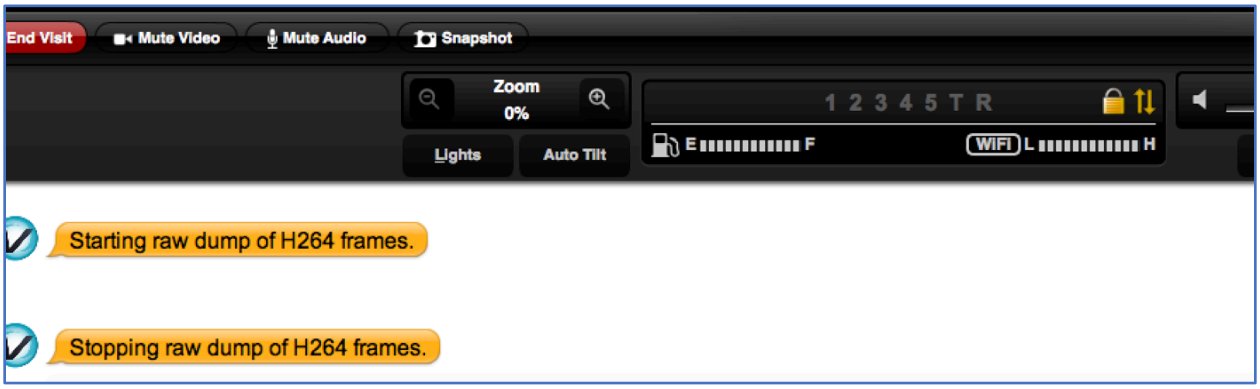

Figure 18: Capturing and stealing live streaming

Finally, we can see the recorded video generated as H264_data.h264 in Figure 19. An attacker can start watching it on his favorite player, violating customer confidentiality!

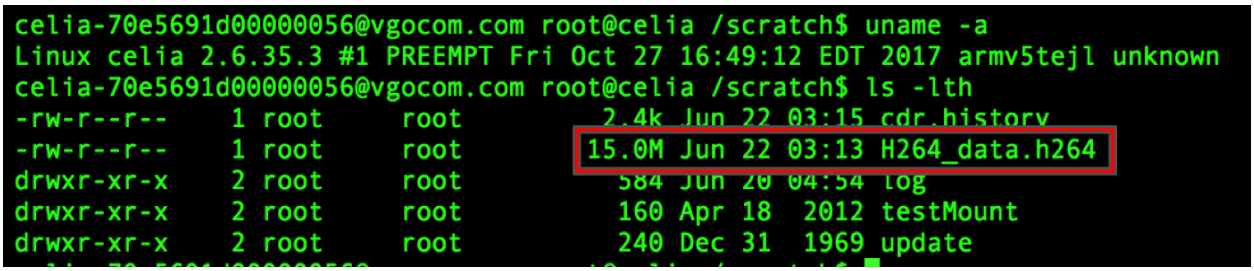

Figure 19: Recorded video generated in the file system

Takeaways

The following are some key lessons we can learn about IoT security from these tests:

- IoT security today is similar to what IT security was in the 1990s. As we learned with IT security, it is important to involve IoT security from the design phase of product development.
- IoT devices need physical interaction with ~~attackers~~ humans by design, so physical security needs to be improved.
- IoT manufacturers and security researchers need to start collaborating more closely. Bug bounty programs are the best approach in this regard. ☺
- You cannot protect what you cannot see, so the immediate step is to identify IoT assets in your network
- Once located, monitor your IoT devices to find anomalies dynamically using technologies like machine learning.
- Trigger real-time alerts so that the security team can take actions immediately and the network devices can react accordingly.