Vacuums in the Cloud
Analyzing Security in a Hardened IoT Ecosystem

Jiska Classen, Johannes Eger, and Matthias Hollick
Secure Mobile Networking Lab - SEEMOO
Technische Universität Darmstadt, Germany

Fabian Ullrich
IT Security Analyst @ ERNW GmbH
Heidelberg, Germany
Motivation

- A vacuum cleaning robot is in your house, has access to your Wi-Fi and knows many of your personal habits.
- **Who of you owns a vacuum cleaning robot?**
- (Own as it's YOUR robot, neither your neighbor's nor someone's on the Internet!)
Motivation

- A vacuum cleaning robot is in your house, has **access to your Wi-Fi** and knows many of your **personal habits**.
- **Who of you owns a vacuum cleaning robot?**
- **Neato** is one of the top vacuum cleaning robot models in the US.
- In Germany, **Vorwerk** has been selling vacuum cleaners forever (founded 1883). Their top model, a robot, is a rebranded Neato.
- Vorwerk won the test comparisons in Germany with their VR300/VR200.
Robots were harmed during our experiments! (Sorry for that...)
No customer data was leaked.
Neato was informed and fixed all issues in time.
Infrastructure & Security Features

UI & USB console
Infrastructure & Security Features

UI & USB console

Manual robot commands
Infrastructure & Security Features

UI & USB console

Manual robot commands

Account information

Setup only

Beehive
Infrastructure & Security Features

UI & USB console

Manual robot commands

Robot commands, status responses

Robot commands / status

Account information

Setup only

Nucleo

Beehive

?(??)
Infrastructure & Security Features

- Secure boot
- Encrypted logs
- Signed firmware updates

Nucleo

- HTTPS
- RSA key (robot→cloud)
- Secret key (robot→user)

Beehive

- HTTPS
- Secret key (robot→user)

Secret key (robot→user)

Nucleo

Secret key (robot→user)

Beehive

HTTPS

HTTPS

HTTPS

RSA key (robot→cloud)

RSA key (robot→cloud)

Secret key (robot→user)

Secret key (robot→user)
Contributions

- With all these security features, what could possibly go wrong?
Contributions

- With all these security features, what could possibly go wrong?

- We bypass secure boot on a vacuum cleaning robot to extract its memory.

- Our key findings are...  
  ...key findings!

- We also gained unauthenticated RCE on robots over the cloud.
Secure Boot Bypass

- Custom *AM335x* chip (guessed by size factor).
- **QNX 6.5 image** from *Foundry27* is bootable but crashes.

- Get QNX SDP, modify image, skip hardware initialization, reboot Neato system into custom image for **cold boot** attack, print all **RAM to the serial port**.

---

Jiska found this, got assigned CVE-2018-20785, fixed in Neato BotVac Connected D7 4.4.0-72 in December 2018 and for Vorwerk somewhen after January 2019.
Secret Key

- Generated when associating a robot with a user account.
- Known by: robot, app and cloud components.
- **Individual** key for each robot/user account relation!
  - Used for authenticating commands to robot.

https://developers.neatorobotics.com/
Secret Key

- Generated when associating a **robot** with a **user account**.
- Known by: robot, app and cloud components.
- **Individual** key for each **robot/user account** relation!
  
  Used for **authenticating commands to robot**.

---

Header = Authorization: NEATOAPP [signature]

1. `string_to_sign = serial + date + message_body`
2. `signature = HMAC_SHA256(secret_key, string_to_sign)`

https://developers.neatorobotics.com/
RSA Key
- Robots have to initially send the secret key
  - Has to be authenticated.
- Secret key not that secret
  - Several third parties know it.
  - Cannot be used to authenticate the robot in the cloud.
  
  **RSA Key** used to authenticate robot to cloud.
Keys and their Purpose (2)

RSA Key

- Robots have to initially send the secret key
  - Has to be authenticated.
- Secret key not that secret
  - Several third parties know it.
  - Cannot be used to authenticate the robot in the cloud.

**RSA Key** used to authenticate robot to cloud.

Header = Authorization: NEATOBOT [serial]:[signature]

1 string_to_sign = serial + http_method + URI + date + body
2 signature = sign_rsa_sha256(string_to_sign, rsa_private_key)
Secret Key Entropy Reduction

```c
    rnd = rand();
    
    time_shift[0:3] = time_now;
    time_shift[4:6] = 0;
    time_shift[7] = 16;
    time_shift[8] = rnd + rnd / 0xFFFF;
    time_shift[9] = entropy_reducing_math(rnd + rnd / 0xFFFF);
    time_shift[10:15] = robot_MAC;
```
Secret Key Entropy Reduction

```c
1   rnd = rand();
2
3   time_shift[0:3] = time_now;
4   time_shift[4:6] = 0;
5   time_shift[7] = 16;
6   time_shift[8] = rnd + rnd / 0xFFFF;
7   time_shift[9] = entropy_reducing_math(rnd + rnd / 0xFFFF);
8   time_shift[10:15] = robot_MAC;
```

```c
3
   return 454;
3
```
Secret Key Entropy Reduction

- Entropy relies on time of robot linkage.
  - One year = 25 bit
  - One hour = 12 bit
- There are multiple offline attack scenarios.

```
1   rnd = rand();
2
3   time_shift[0:3] = time_now;
4   time_shift[4:6] = 0;
5   time_shift[7] = 16;
6   time_shift[8] = rnd + rnd / 0xFFFF;
7   time_shift[9] = entropy_reducing_math(rnd + rnd / 0xFFFF);
8   time_shift[10:15] = robot_MAC;
```

```python
rand () 3
return 454;
```
RSA Keys for Robot Authenticity (1)

- Encrypted RSA keys in /var/keys.
- `vendorPrivateKeyProduction` sounds promising!
- Let’s do some string de-obfuscation!
RSA Keys for Robot Authenticity (1)

- Encrypted RSA keys in /var/keys.
- `vendorPrivateKeyProduction` sounds promising!
- Let’s do some string de-obfuscation!

- **RSA key** is the **same** for **all robots**.
RSA Keys for Robot Authenticity (1)

- Encrypted RSA keys in /var/keys.
- `vendorPrivateKeyProduction` sounds promising!
- Let’s do some string de-obfuscation!

- **RSA key** is the same for all robots.

![Diagram showing robot authenticating with a `I'm a robot!` message and a `Okay :)` response.](image-url)
RSA Keys for Robot Authenticity (2)

- We are able to **impersonate arbitrary robots**.
  - Allows for multiple other attacks.
  - For example: **Leak victim’s smartphone IP**
We are able to **impersonate arbitrary robots.**
- Allows for multiple other attacks.
- For example: **Leak victim’s smartphone IP**

---

**RSA Keys for Robot Authenticity (2)**

- We are able to **impersonate arbitrary robots.**
- Allows for multiple other attacks.
- For example: **Leak victim’s smartphone IP**

---

**Diagram:**

- **Hacker**
- **Nucleo**
- **Robot location?**
- **Does no longer get requests**
We are able to **impersonate arbitrary robots.**
- Allows for multiple other attacks.
- For example: **Leak victim’s smartphone IP**
RSA Keys for Robot Authenticity (2)

- We are able to **impersonate arbitrary robots**.
  - Allows for multiple other attacks.
  - For example: **Leak victim’s smartphone IP**.
We are able to **impersonate arbitrary robots**.
- Allows for multiple other attacks.
- For example: **Leak victim’s smartphone IP**
Local_10 = fopen(acStack120,"rb");
if (local_10 != (FILE *)0x0) {
    sVar2 = fread(local_c,local_11c,1,local_10);
    if (sVar2 != 0xffffffff) {
        fclose(local_10);
        rc4_ksa("*^JEd4WI",9,0);
        ...
    }
}

- Binary /bin/rc4_crypt is called without arguments and contains a hardcoded key.
- Coredumps contain the complete binary that crashed (/bin/robot, /bin/astro, ...)
  - ...get binaries from robots without secure boot bypass!
- Yes, even /bin/robot can crash!
  - ...got a /bin/robot coredump from a non-connected Neato Botvac 85 (SW 1.2.1)

Jiska found this and got assigned CVE-2018-17177.
Unauthenticated RCE

- **Buffer overflow** in Nucleo cloud connection daemon.

Fabian found the crash and performed analysis, Jiska decrypted the coredump, Johannes did further analysis and implemented a PoC (CVE-2018-19442).
Unauthenticated RCE

- **Buffer overflow** in Nucleo cloud connection daemon.
- Can be triggered with requests to `https://nucleo.neatocloud.com:4443/vendors/neato/robots/[robot_serial]/messages`.
- The overflow is within parsing the authentication header, which means that we found an **unauthenticated RCE**!
- All services run as **root**.
- Fix: Authentication headers are validated on Nucleo.

Fabian found the crash and performed analysis, Jiska decrypted the coredump, Johannes did further analysis and implemented a PoC (CVE-2018-19442).
Security Implications (1)

- **IoT product at home? Keep it offline!**

- As a **customer:**
  - **Update** your robot.
  - **Hide your robot’s serial number!**
Security Implications (2)

- Connected ecosystem developers:
  - Using RSA, RNG, hashing, secure boot, encrypted logs, signed firmware updates sounds good...
  - Review cryptographic key components and root of trust assumptions.
  - **Dissecting one of your products** should not compromise security of the other products, i.e., similar keys.
  - Test your security relevant code in practice to uncover issues like the static secret key “random” function—check entropy before hashing.
Security Implications (3)

- Lessons learned during **responsible disclosure**:
  - Scientifically **proven secure** methods are insecure if applied wrong.
  - Normal **developers** are not aware of ASLR, DEP, ..., they **just write code**.
  - Lack of easy to understand **guides** of how to build connected infrastructures.
    - Too many guides?
    - No reasoning why certain measures are needed in an infrastructure.
  - Penetration testing often does not cover specific **implementation details**.
    - Web API only tests...