STARfoods

Food that's out of this world!

An introductory Cyber-Physical systems and IoT cybersecurity challenge.

Scenario

Welcome to STARfoods!

Welcome to STARfoods, a leading innovator in the global food industry. As a forward-thinking company, STARfoods prides itself on pushing the boundaries of technology and sustainability to revolutionize how food is produced, packaged, and delivered. As part of our continued commitment to excellence, we are embarking on an ambitious project to outfit our state-of-the-art headquarters with cutting-edge IoT devices.

Unlike other companies that rely on third-party products, STARfoods is confident in our ability to design, develop, and deploy our own in-house IoT solutions. This decision is driven by our belief that we can tailor these devices to meet our specific needs, enhance operational efficiency, and maintain the highest standards of security.

As part of our commitment to the community and to nurture the next generation of cybersecurity professionals, STARfoods is excited to announce a unique competition for students studying cybersecurity. We're offering you the chance to put your skills to the test by evaluating the security of our newly developed IoT devices and infrastructure. This is not only a rare opportunity to work with real-world systems but also a chance to contribute to the security of a global industry leader.

The stakes are high: the student who identifies the most vulnerabilities and provides the most effective solutions will be rewarded with a Titanium STARfoods Rewards account, granting them free STARfoods products for life! While this is our way of giving back to the community, we also recognize that the insights from this competition will help us refine our systems and push our technological boundaries even further.

Are you ready to take on this challenge? Your expertise could lead you to a lifetime of STARfoods products and a place in the history of our company's technological evolution.

Best regards,
Shay Savory
CEO, STARfoods

Implementation

In an effort to pursue high-security solutions and modernize our infrastructure, we've opted to write all STARfoods Smart Devices in Rust. For this event, we've provided you an Ubuntu 22.04 virtual machine with the compiled software for our Smart Devices. The software will be located on the "star" user's desktop within the "STARfoods " folder (/home/star/Desktop/STARfoods).

Within the STARfoods folder, there will be a "devices" folder, in which there will be six executables, each representing a different smart device. To interact with any of these executables, you simply can execute them within the "devices" directory. For example, to interact with the "smart_light", you can simply run `./smart_light` in the command line.

Submission Instructions

The "STARfoods" event allows for a maximum of 10 submission attempts, with all submissions required to be submitted in CSV format. In adherence to the format illustrated in Figure 1 below.

```
<challenge_name>,<answer_1>,<answer_2>,<answer_3>,<answer_4>,<...>,<answer_n>
```

Figure 1

Challenges are defined as all questions and scenario items associated with a specific “Smart Device.” Challenges consist of multiple *challenge questions*.

Submission entries are defined as each line in the submitted CSV that follows the format outlined in Figure 1. Provide your answers in the order corresponding to the challenge questions. Each challenge will have between 1 and *n* challenge questions, where *n* is a positive integer.

The “Challenge Name” field must be appended to the beginning of each submission entry. The challenge name indicates which “Smart Device” your answers correspond to. For example, if you are working on the questions for the “smart_light” device, a submission entry may look like the example in figure 2.

```
smart_light,SomeAnswer,AnotherAnswer,AThirdAnswer,MoreAnswers,AnAnswer
```

Figure 2

Each challenge question will be underlined and marked with a *question number*. Submit challenge questions in order of their question number. An example of two challenge questions with their corresponding answers (highlighted) associated with an example challenge “smart_door” is seen in figure 3 below.

1-What is required to change the door status? (EX: username)
token

2-What is the name of the Smart Device? (EX: STARTOKEN)
ASTRALDOOR

Figure 3

A submission entry consisting of the above challenge questions for the “smart_door” challenge would look like the entry in figure 4.

```
smart_door,token,ASTRALDOOR
```

Figure 4

For all answers to challenge questions, assume that you are working in the same directory as the binary corresponding to the challenge. Meaning that to execute each binary and it’s arguments it looks like `./binary_name option1 option2`

Challenges

Each challenge consists of the following steps to help guide your analysis. Each step will contain one or more corresponding questions. The steps are as follows:

- **Interaction**: Interact with the software to understand how it works
- **Review**: Review a portion of the source code and identify the vulnerability
- **Exploit**: Exploit the vulnerability found to either gain hidden information or disrupt the software.

The challenges are as follows:

- smart_light
- smart_lock
- smart_token
- smart_thermostat
- smart_fridge
- smart_purifier

Challenge descriptions will begin on the next page of this document. We thank you for your contribution to STARfoods and the cybersecurity community as a whole.

SMART_LIGHT

We've recently installed a “smart light” in the office of one of our senior employees. Please investigate this smart light for vulnerabilities by answering the following questions below.

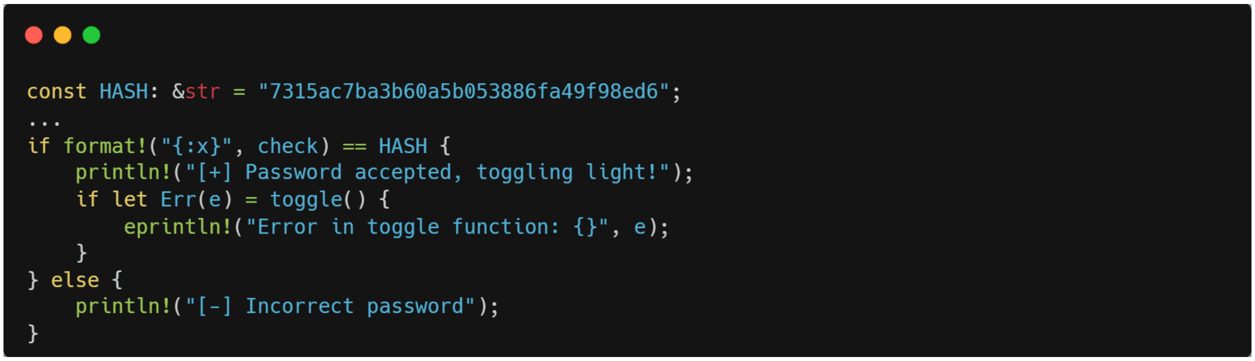
INTERACTION

Try turning on the smart light by running the `smart_light` binary.

1-What is required to change the light status? (EX: username)

REVIEW

The program uses the following hardcoded password as seen in the code snippets below:



```
const HASH: &str = "7315ac7ba3b60a5b053886fa49f98ed6";
...
if format!("{:x}", check) == HASH {
    println!("[+] Password accepted, toggling light!");
    if let Err(e) = toggle() {
        eprintln!("Error in toggle function: {}", e);
    }
} else {
    println!("[-] Incorrect password");
}
```

2-What is the ID of the MITRE ATT&CK technique is this device vulnerable to?

EXPLOIT

3-What is the password needed to change the light status?

4-What is the current light status? (on/off)

5-What can you run on the command line to change the light status?

6-What is the name of the smart light device?

SMART_LOCK

We've installed a smart lock that controls the doors to our food product research labs' pantries. To unlock a pantry, you simply have to enter in an access code, and the pantry corresponding to that code will open. Each pantry has a research team assigned to it. Please analyze the smart lock software, identifying any vulnerabilities along the way, and answer the following questions:

INTERACTION

Run the smart lock software with the following command: `./smart_lock`

1- What option (EX: username) is required to operate the software?

2-Which pantry (EX: Pantry X) is opened with code 9191?

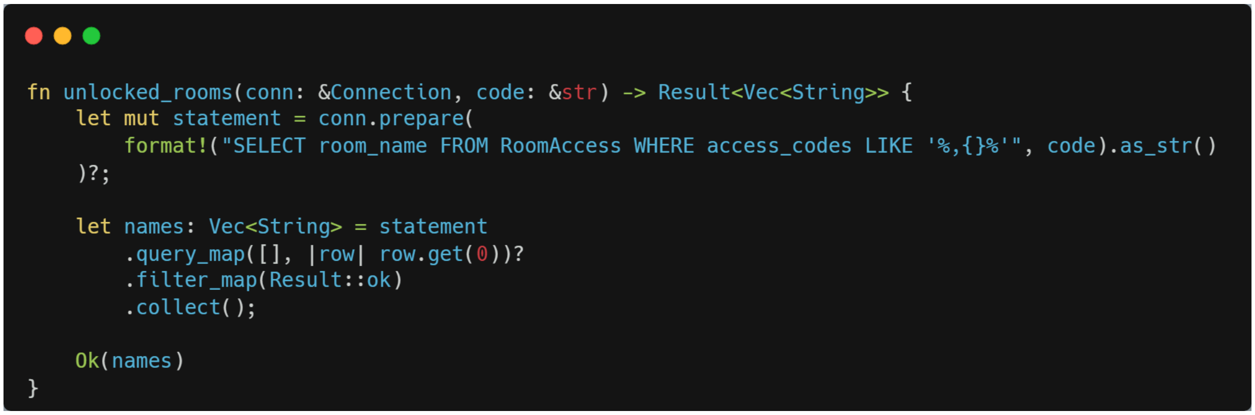
3-What team (EX: Team Xenon) is assigned to the pantry that is unlocked with code 9191?

4-Which pantry (EX: Pantry X) is opened with code 3342?

5-What team (EX: Team Xenon) is assigned to the pantry that is unlocked with code 3342?

REVIEW

Below is the function responsible for returning the unlocked rooms:



```rust
fn unlocked_rooms(conn: &Connection, code: &str) -> Result<Vec<String>> {
    let mut statement = conn.prepare(
        format!("SELECT room_name FROM RoomAccess WHERE access_codes LIKE '%,{}%'", code).as_str()
    )?;

    let names: Vec<String> = statement
        .query_map([], |row| row.get(0))?
        .filter_map(Result::ok)
        .collect();

    Ok(names)
}
```

6-What is the acronym for the vulnerability this program is vulnerable to? (EX: XSS)

7-What is the ID of the technique in the MITRE ATT&CK Framework that is often associated with the vulnerability in this code?

EXPLOIT

8-What is the team that is supposed to tentatively use the newest pantry?

SMART_TOKEN

We've heard good things about multi-factor-authentication and have decided to implement a similar system. We've built a secure MFA verification software that employees will be able to use to sign into their accounts and even physical spaces in the future. Please analyze the token software, identifying any vulnerabilities along the way, and answer the following questions:

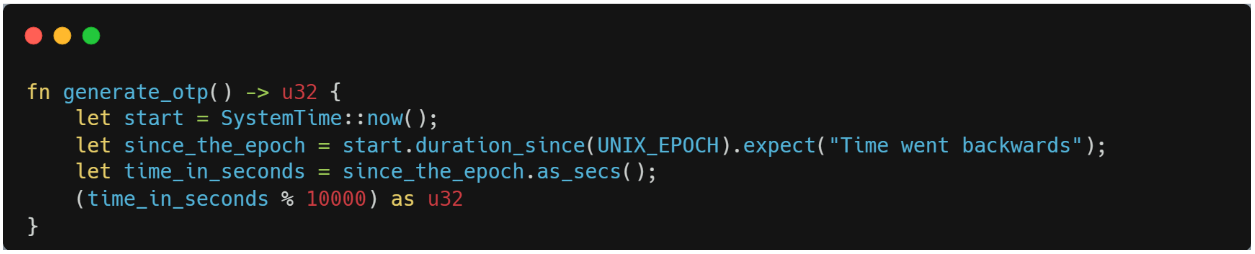
INTERACTION

Run the smart lock software with the following command: `./smart_token`

1-What is the name of the MFA device? (EX: Quasar PC)

REVIEW

Below is the function responsible for MFA code generation.



```
fn generate_otp() -> u32 {
    let start = SystemTime::now();
    let since_the_epoch = start.duration_since(UNIX_EPOCH).expect("Time went backwards");
    let time_in_seconds = since_the_epoch.as_secs();
    (time_in_seconds % 10000) as u32
}
```

2-What format is a valid token in (character type and number)? (EX: hexadecimal:12 for 12 hexadecimal characters)

EXPLOIT

3-What is the serial number of the device?

SMART_THERMOSTAT

We started installing THERMOSTAR smart thermostats across the building. There is a central control panel that allows us to manage all of them at once. Please analyze the smart thermostat control panel software, identifying any vulnerabilities along the way, and answer the following questions:

INTERACTION

Run the smart lock software with the following command: `./smart_thermostat`

1-How many thermostats are displayed by default?

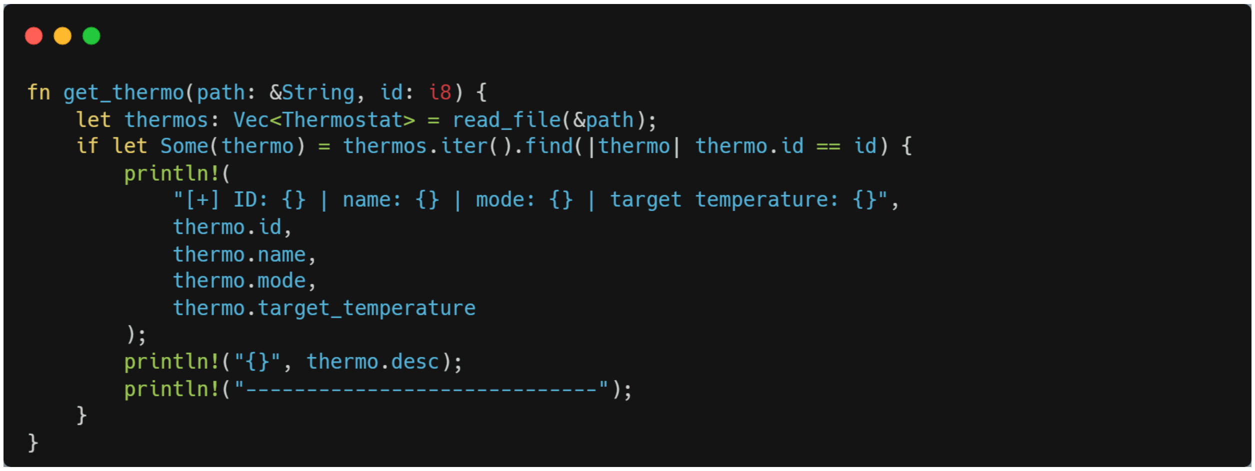
2-What is the name of the thermostat with an ID of 2?

3-What command would you run to switch the mode (heating/cooling) of a thermostat with an ID of 1?

4-What command would you run to set the target temperature to 74 degrees for a thermostat with an ID of 1?

REVIEW

Below is the function responsible for MFA code generation.



```rust
fn get_thermo(path: &String, id: i8) {
    let thermos: Vec<Thermostat> = read_file(&path);
    if let Some(thermo) = thermos.iter().find(|thermo| thermo.id == id) {
        println!(
            "[+] ID: {} | name: {} | mode: {} | target temperature: {}",
            thermo.id,
            thermo.name,
            thermo.mode,
            thermo.target_temperature
        );
        println!("{}", thermo.desc);
        println!("-----------------------------");
    }
}
```

5-What is the acronym for the vulnerability this program is vulnerable to? (EX: XSS)

EXPLOIT

6-What is the name of the thermostat that is not displayed by default?

7-What is the current target temperature of the thermostat that is not displayed by default?

SMART_FRIDGE

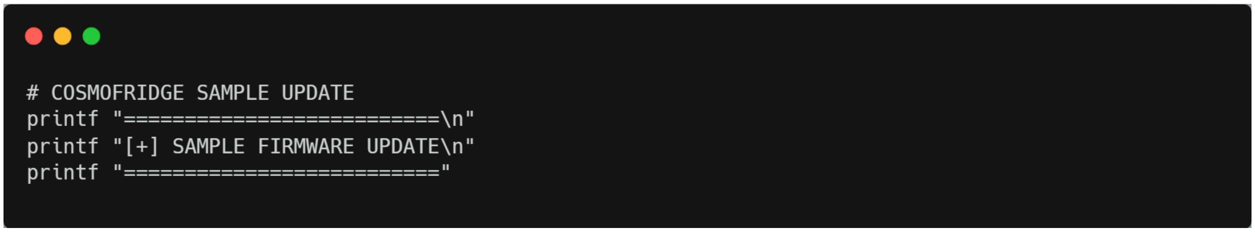
We just acquired a new smart fridge for our employee break room. It tells you what's inside without you having to check yourself! Please analyze the fridge software, identifying any vulnerabilities along the way, and answer the following questions:

INTERACTION

Run the smart lock software with the following command: `./smart_fridge`

1- How many cookie dough ice cream tubs are in the smart fridge?

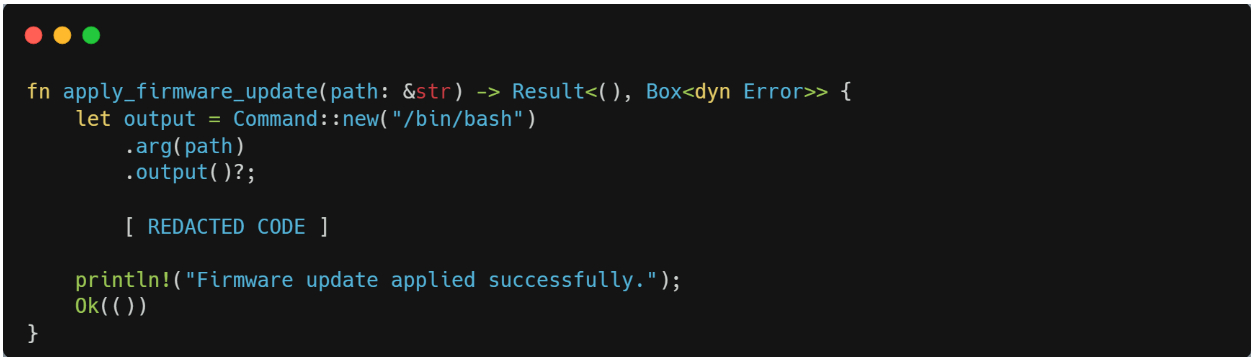
We were told that you can run firmware updates via “.update” files. Here is a sample .update file we were given.



```
# COSMOFRIDGE SAMPLE UPDATE
printf "==========================\n"
printf "[+] SAMPLE FIRMWARE UPDATE\n"
printf "=========================="
```

REVIEW

Below is the function responsible for updating the firmware.



```
fn apply_firmware_update(path: &str) -> Result<(), Box<dyn Error>> {
    let output = Command::new("/bin/bash")
        .arg(path)
        .output()?;

        [ REDACTED CODE ]

    println!("Firmware update applied successfully.");
    Ok(())
}
```

2-What is the ID of the technique in the MITRE ATT&CK Framework that most closely aligns with the vulnerability in this code?

EXPLOIT

3-What is the “secret”?

SMART_PURIFIER

We're building a new smart purifier and planning to install it in our COO's office. Please analyze the purifier's software, identifying any vulnerabilities along the way, and answering the following questions.

INTERACTION

Run the smart lock software with the following command: `./smart_purifier`

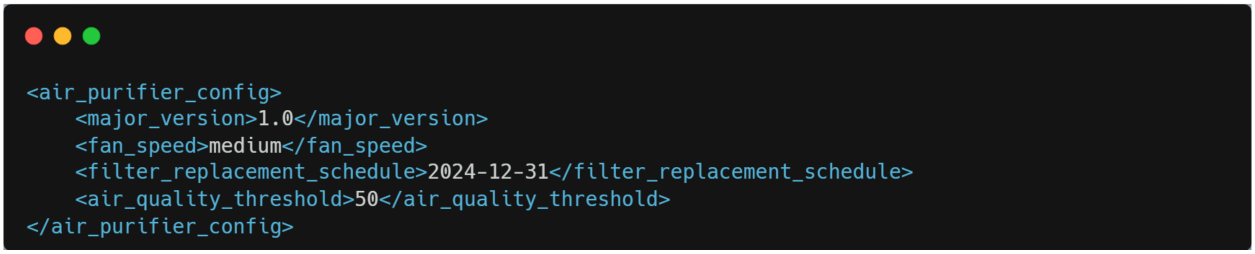
1- What is the name of this model of air purifier?

2-When is the filter scheduled to be replaced?

3-What is the current fan speed?

4-What is the major version of the purifier?

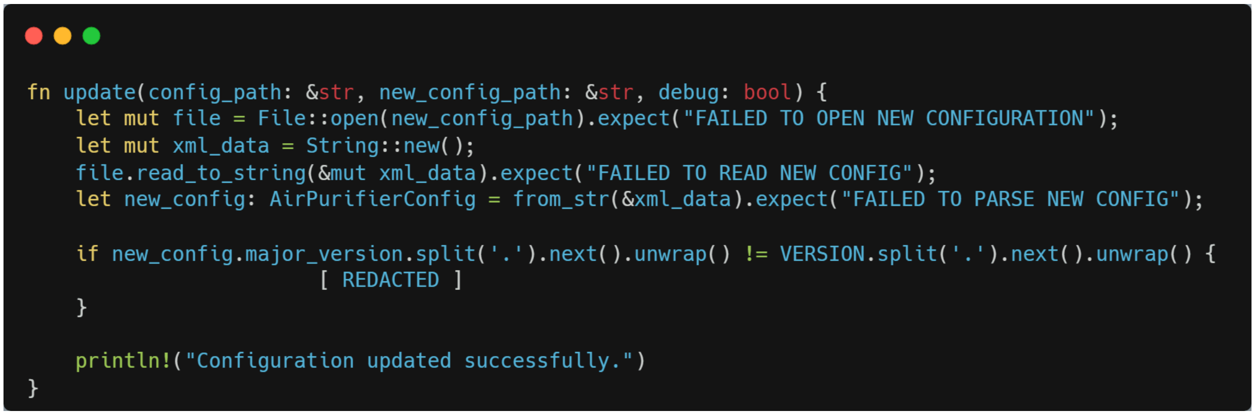
The team that built this software didn't do much documentation, but from our understanding you can update the configuration using a file with contents similar to the example below.



```xml
<air_purifier_config>
    <major_version>1.0</major_version>
    <fan_speed>medium</fan_speed>
    <filter_replacement_schedule>2024-12-31</filter_replacement_schedule>
    <air_quality_threshold>50</air_quality_threshold>
</air_purifier_config>
```

REVIEW

Below is the function responsible for updating the configuration.



```rust
fn update(config_path: &str, new_config_path: &str, debug: bool) {
    let mut file = File::open(new_config_path).expect("FAILED TO OPEN NEW CONFIGURATION");
    let mut xml_data = String::new();
    file.read_to_string(&mut xml_data).expect("FAILED TO READ NEW CONFIG");
    let new_config: AirPurifierConfig = from_str(&xml_data).expect("FAILED TO PARSE NEW CONFIG");

    if new_config.major_version.split('.').next().unwrap() != VERSION.split('.').next().unwrap() {
                    [ REDACTED ]
    }

    println!("Configuration updated successfully.")
}
```

5-What option can help us reveal data that might be unintended for end users in a production setting?

6-What is the full version of the air purifier software?