Safety report linked to the innovative project

The aim of this report is to understand a scientific article that enlights a weakness and/or a security solution and to explain it.

1. Context

The project aims at developing safety support tools for firefighters in buildings subject to dangerous hazards such as a fire.

We have developed IoT devices equipped with temperature, sound and gas sensors as well as a connected watch. These sensors send their data over WIFI in HTTP format to a server. The connected watch can receive information in the form of an alert in the event of temperatures exceeding a threshold, for example.
Sensors are built with ESP8266 micro-controller. The code is developed in C language with Arduino IDE.

2. Actual network architecture

The project network architecture's is as shown below :

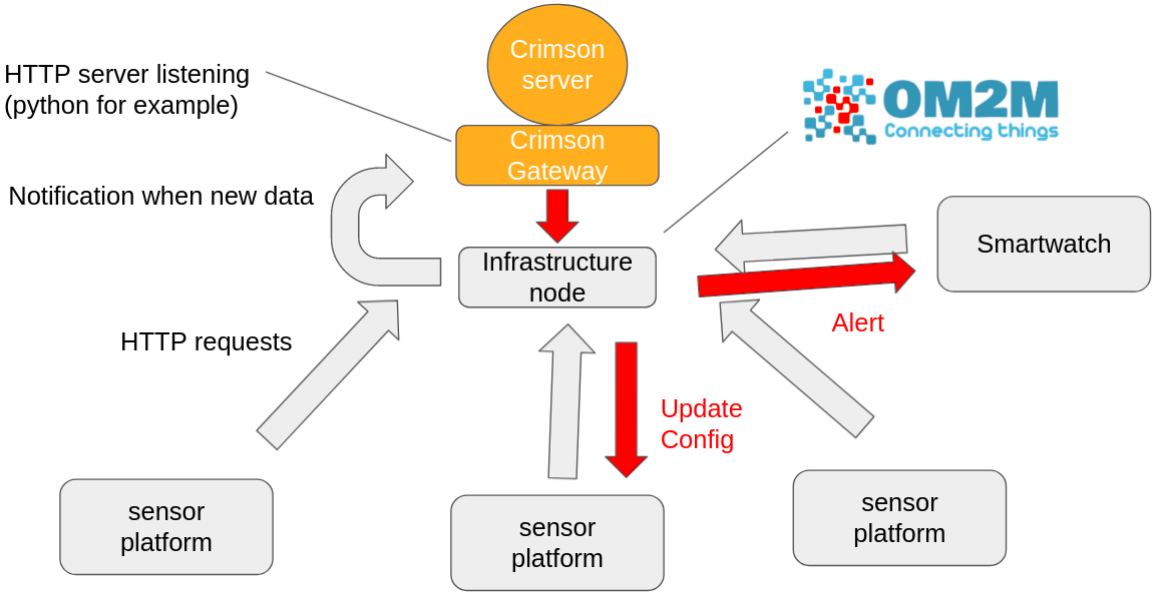

Fig.1 : Basic project architecture

Our application requires the use of many different objects. First of all, it needs to collect various data on the internal environment of the building, which implies the use of many different sensors. We can have temperature sensors, gas sensors, sound sensors, obstacle sensors, among others.

Each of these sensors may have a different protocol and communication standard and thus make the interoperability of the nodes complex. In addition, other types of objects such as connected watches increase the complexity of setting up such a system. Finally, hypervision for the management of critical interventions requires the use of many tools, and not only the system we want to develop.

Interoperability between the different tools is also a crucial point in order to allow their perfect collaboration and thus improve the value generated. It is partly for these reasons that we chose to use an IoT standard. As the project is at the beginning of its development, the technological and technical choices made to date are likely to be improved or changed in the future. Setting a standard also makes it possible to integrate these changes by abstracting certain layers such as the communication layer for example. We have therefore chosen the oneM2M[1] standard implemented with the OM2M[2] platform developed by the Laas CNRS and the Eclipse foundation.

The terminal nodes are the different sensors, the connected watches and the server on which the 3D model of the building is loaded. The data acquired by the sensors is transferred to the infrastructure node by Wi-Fi, the core of our architecture, through http requests. The sensors themselves may not support the http protocol. In this case it may be necessary to develop an Interworking Proxy Entity (IPE) specific to the sensor's communication technology (the sensor technology in general) to ensure its integration into the system. For the time being, we ignore the communication protocol on which our system no longer depends thanks to the possibility of easily integrating IPE.

Once the data is sent to the infrastructure node, an entity of the oneM2M standard, it is automatically sent to the server through a notification and subscription system. We have also represented a downlink communication from the server to the sensors, for example to update the configurations (such as the sampling rate) of the sensors, but this link is not implemented for the moment.

[1] https://www.onem2m.org/
[2] https://www.eclipse.org/om2m/

In this security report we focus on Wi-Fi security vulnerabilities.

To illustrate the work done, we can see below the Wi-Fi sensor embedded code:

```
// WIFI params for IoTs communication
char* WIFI_SSID = "GalaxyS20"; // Configure here the SSID of your WiFi Network
char* WIFI_PSWD = "zxla4008"; // Configure here the PassWord of your WiFi Network
int WIFI_DELAY  = 100; //ms
```

```
//WiFi init
void init_WiFi() {
  Serial.println("Connecting to  " + String(WIFI_SSID) + " ...");
  WiFi.persistent(false);
  WiFi.begin(WIFI_SSID, WIFI_PSWD);

  // wait until the device is connected to the wifi network
  while (WiFi.status() != WL_CONNECTED) {
    delay(WIFI_DELAY);
    Serial.print(".");
  }

  // Connected, show the obtained ip address
  IP_sensor=WiFi.localIP().toString();
  Serial.println("WiFi Connected ==> IP Address = " + IP_sensor);

```

Fig.2 : ESP8266 embedded code

As we can see, there is just securisation by a login with a password, both are directly sent to the gateway without encryption.

3. Objectives of potential attackers

- Server saturation
- Firefighter endangerment via denial of services

4. Capacity of attackers

- Physical probe access
- Hacking the WIFI network between the watch and the server (man in the middle)
- Power injection
- WIFI jamming
- Direct server attack

5. Identification of possible vulnerabilities

- Falsification of the values sent by the probes to the server
- Simulation of additional probes or dummy watches in order to overload the server
- Falsification of the values sent by the watch to the server

6. Different attacks and proposed solutions

By the reading of the Usenix scientific article chosen [1], we found some vulnerability to correct in our project.
Authors write about the frame aggregation functionality and the frame fragmentation functionality. These functionalities have design flaws that enable a malicious user to forge encrypted frames in various ways, which in turn enables exfiltration of sensitive data.
In their experiments, all devices were vulnerable to one or more of their attacks, confirming that all Wi-Fi devices are likely affected.

Different attacks can be done as below:

1. Spoofing aggregated frames

The spoofing consists of sending IP packets from a source IP address that has not been assigned to the computer sending them. This design flaw has been assigned CVE-2020-24588.
In our project, an attacker can send some packet in the building to send bad information to the smartwatch.
The design flaw is in the frame aggregation feature of Wi-Fi. This feature increases the speed and throughput of a network by combining small frames into a larger aggregated frame. To implement this feature, the header of each frame contains a flag that indicates whether the (encrypted) transported data contains a single or aggregated frame. This is illustrated in the following figure:

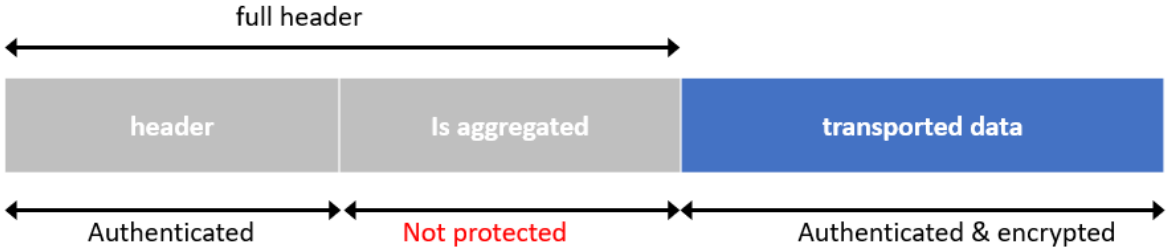

Fig. 3 aggregated frame

But this "is aggregated" flag is not authenticated and can be modified by the attacker, meaning the smartwatch can be tricked into processing the encrypted transported data in an unintended manner. An attacker can abuse this to inject arbitrary network packets by tricking the smartwatch into connecting to their server and then setting the "is aggregated" flag of carefully selected packets. In the study, all tested devices were vulnerable to this attack. The ability to inject packets can in turn be abused to intercept a victim's traffic by making it use a malicious DNS server.
This design flaw can be fixed by authenticating the "is aggregated" flag. The Wi-Fi standard already contains a feature to authenticate this flag, namely requiring SPP A-MSDU frames, but this defense is not backwards-compatible and not supported in practice. Attacks can also be mitigated using an ad-hoc fix, though new attacks may remain possible.

2. Mixed key attack

This design flaw comes from the process of fragmentation of a frame, which is performed when a frame is too long to be efficiently transmitted and thus needs to be split into fragments transmitted and acknowledged independently.

When a frame is fragmented, each fragment is encrypted with the same key, but the receiver does not require the fragments to be decrypted by the same key to be reassembled. The mixed key attack abuses that design flaw by forwarding or withholding specific fragments encrypted with different keys. By doing so an attacker can forge a frame by reassembling fragments decrypted with different keys.

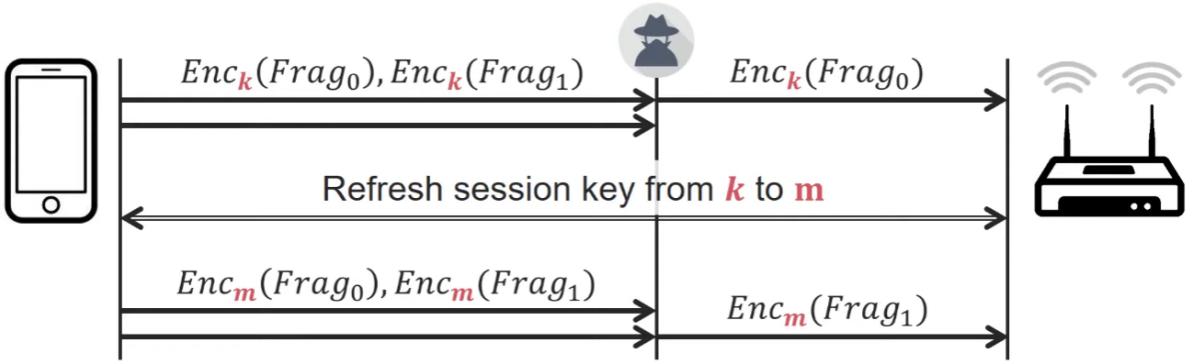

Fig.4 mixed key attack

To perform this attack, the adversary cautiously chooses which fragment(s) to forward or withhold from fragmented frames he receives. Moreover, he forwards specific fragment(s) from different frames before and after a session key has been refreshed, as shown in Fig.4. By doing so, the fragments sent before and after the refreshing of the session key are reassembled as a single frame. This vulnerability can be exploited to forge a corrupted frame that will be accepted by a receiver.

This attack works against the following protocols: WEp, CCMP, and GCMP, and is only possible under the following specific requirements.
First, the device(s) needs to send fragmented packets, which is only recommended in noisy environments unless with Wi-Fi 6 where dynamic fragmentation is often set by default. Therefore, recent devices which use more and more Wi-Fi 6 are likely to be more vulnerable to this attack. Secondly, the victim needs to be connected to the server of attackers. And finally, the network has to periodically refresh the session key.

To fix this vulnerability, we would need to properly isolate security contexts. In particular, we should properly separate data that is decrypted with different keys so that each fragment can only be reassembled with fragments from the same original frame.

3. Cache attack

The cache attack works by using connection to a hotspot network (Like Eduroam) or to an enterprise network, a network where devices don't trust each other. The attacker logs in with real credentials, and then spoofs the victims mac address so that the fragments we leave behind will be saved under the victims address. This is an important point since the fragments are stored on mac addresses and not the login credential.

This attack utilizes the fact that fragments are not cleared after reconnecting, which here is considered a design flaw because therefore the attacker can poison the cache.
Fig.5 shows how an attacker could connect and poison the first fragment, then disconnects and waits until the victim connects. When the victim sends something it will be the attacker's fragment that will be the first part of the message the access point reads.

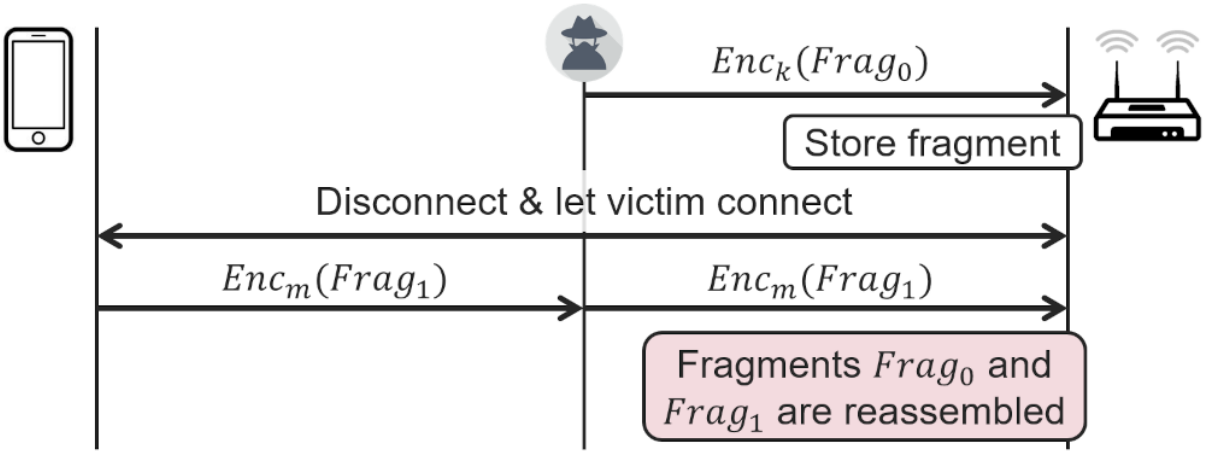

Fig.5 Cache attack

The attack works on the following protocols: WEP, CCMP and GCMP, and it also requires devices to be sending fragmented frames just like in the mixed key attack. On the test performed in this report they found that around half of the devices were vulnerable to this attack.
It is not unlikely that we would have devices connecting to these types of access points if this would be a widespread adoption of our IoT device. Therefore it would be wise to look into how we could prevent this. There is proposed an easy fix to this attack which is simply to always delete fragments from the cache when a device disconnects. Moreover I would also recommend that all of our devices are not connected to a hotspot that normal people have access to

7. References

[1] Fragment and Forge: Breaking Wi-Fi Through Frame Aggregation and Fragmentation
https://www.usenix.org/conference/usenixsecurity21/presentation/vanhoef

[2] CVE-2020-24588 – Principe Aggregation Attack – #Fragattacks
https://www.fragattacks.com/#usenixpres
https://papers.mathyvanhoef.com/fragattacks-slides-2021-03-8.pdf