Data Exfiltration Via Smart Plugs

*Hackfest 2020 : IoT Hacking Challenge
*Team: The Emperors
*Authors:
**Mohamed Ali Belhajali
**Mohamed Aymen Karmous

***Abstract*—With the expansion of the Internet of Things (IoT), connected devices reached 38.5 billion devices in 2020[20c]. And as manufacturers are competing on who would create an IoT based device and sell it to the customers, most of them are ignoring the security issues related to the privacy, the availability of the devices even after getting warned from the security researchers. In this report, we will be demonstrating that covert channel communications can be used to exfiltrate data from an air gaped network using a smart plug based on the Hue and the Saturation variation from the HSV (Hue, Saturation, Value/Brightness) color representation.**

I. Introduction

Smart plugs are widely used to control whatever device you plugged in even when you're not in the room or in the house. This can be done using an application that interacts with the smart plug.
Recent cyber attacks are getting more dangerous by getting control of IoT devices. It is possible to launch any cyber attack (like a DDoS attack) using an IoT botnet or to infiltrate the network through an IoT device or to exfiltrate any data from it.

II. Preliminaries

A. Data Exfiltration

Data exfiltration is a form of data theft that occurs after getting inside a compromised system to steal valuable data. It affects the data privacy. Data exfiltration occurs into the "Actions on Objective" stage of the Cyber Kill Chain model (see Fig. 1). This means that we will focus on how the data exfiltration is done and not how to compromise the system from which the data will be exfiltrated or what data should we exfiltrate.

B. Covert Channel

A Covert Channel is a type of attack that creates a way of communicating through a system for which it was not designed or intended for such communication. That's why it is one of the main techniques used by attackers to steal data from victims.[19]

C. Smart plug operation

In this study, the smart plug uses WiFi norm (2.4GHz IEEE802.11 b/g/n). We will be working on "Maxcio smart plug" (see Fig. 2).

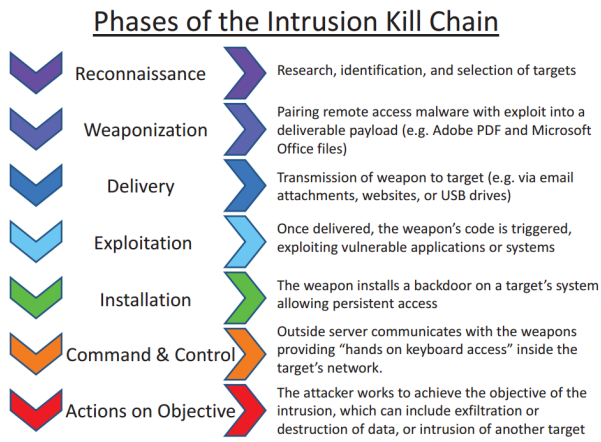

Fig. 1. Cyber Kill Chain stages[20k]

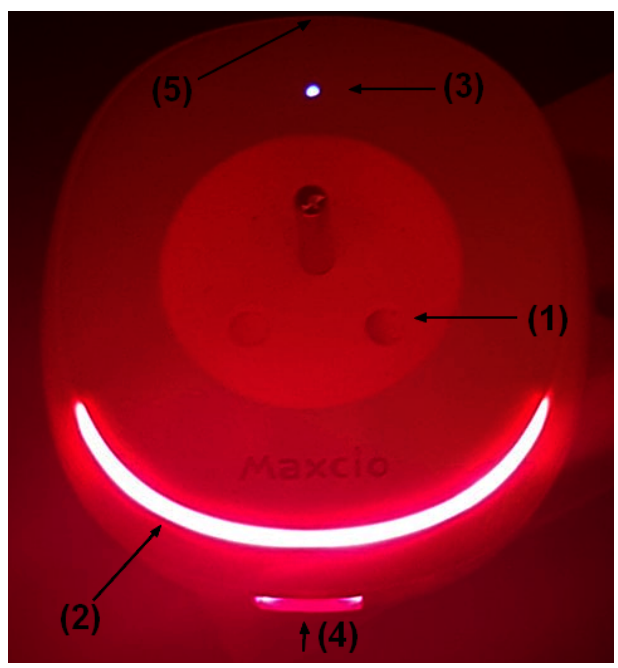

Fig. 2. Smart plug

The smart plug is used as an intermediate between the electrical outlet (that provides electricity) and an electronic outlet (that needs electricity) (see arrow number 1 on Fig. 2). The smart plug has a lamp that works in 3 modes: a white mode, a color mode (with a fixed color) and a scene mode (with a dynamic color) (see arrow number 2 on Fig. 2).
The smart plug can stop providing electricity to the plugged electronic device. When the blue LED (see arrow number 3 on Fig. 2) is on, the electricity is provided. Otherwise, there is no electricity for the plugged electronic device.
The smart plug can share electricity from the USB port (see arrow number 4 on Fig. 2).
The smart plug can be reset using the reset button (see arrow number 5 on Fig. 2).
The smart plug needs to be connected to internet using WiFi to send the data to the cloud. The SSID and its keyphrase are communicated to the smart plug through the "Smart Life - Smart Living" mobile application [20i]. This means that the entry point could require to download and reverse this application to understand how the communication between the smartphone and the smart plug is performed.
The idea about how to exfiltrate the data is to hack the wiFi, that is used by the smart plug, and to communicate with the smart plug from a compromised system to exfiltrate the data. The data exfiltration will be done using the Hue and Saturation variation from the HSV color representation after encoding the data, that should be exfiltrated, into bits and after sending it to the smart plug. This means that the hacker should have a camera focused on the smart plug that records the color and light variation which defines the covert channel attack.

III. Environment preparation

In this section, we will be discussing about the steps that needs to be done before starting the data exfiltration.

A. Discovering the smart plug's mobile application

Discovering the smart plug's mobile application is an easy way to have a general idea what the mobile application can do with the smart plug.

We downloaded the latest version of "Smart Life - Smart Living" mobile application [20i] and we installed it. The first thing that should be done when we try to discover the smart plug using WiFi is to provide the SSID and its password that should be used by the smart plug. We have also to reset the smart plug using the reset button (see arrow number 5 on Fig. 2). The blue led (see arrow number 3 on Fig. 2) will start blinking which means that the smart plug is ready to receive the SSID and its password. Then, the smart plug will be ready to be added in the mobile application with a configurable name. The Fig. 3 shows what the user can do with the smart plug from the mobile application. He can choose the mode (see arrow number one), change the color in the color picker (see arrow number two), change the saturation (see arrow number three) and change the brightness (see arrow number four).

Fig. 3. Smart Life application preview

B. Reverse engineering the mobile application

To understand how the communication between the smartphone and the smart plug is technically done, it's important to reverse engineer the "Smart Life - Smart Living" application by downloading the APK (Android Package Kit) file from a third party website by providing the Google Play link. In this study, we used APKPure website[20j].
The first time, we downloaded the latest version (v3.15.1) we found that the mobile application is communicating with the smart plug using the Smart Config mechanism dedicated for ESP devices[20e] to transmit the WiFi SSID and keyphrase[13]. This has been confirmed using the network protocol analyzer Wireshark[20m] (see Fig. 4).

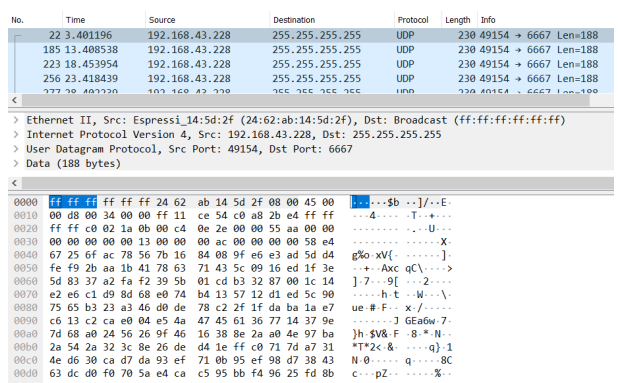

Fig. 4. Wireshark Traffic Capture After Plugging a Smart Plug Configured With an SSID And a Keyphrase

The packet's content is irrelevant since SmartConfig works encode the information on the size of the packets sent (See

reference [13] for more technical details).
So, the SSID and its keyphrase are sent to the smart plug once the first time the smart plug will be configured[20l].

C. Communication with the smart plug

When the mobile application sends the SSID and its keyphrase the device stay in the smart config mode (listening to the packet exchange on the air) (see Fig. 5).

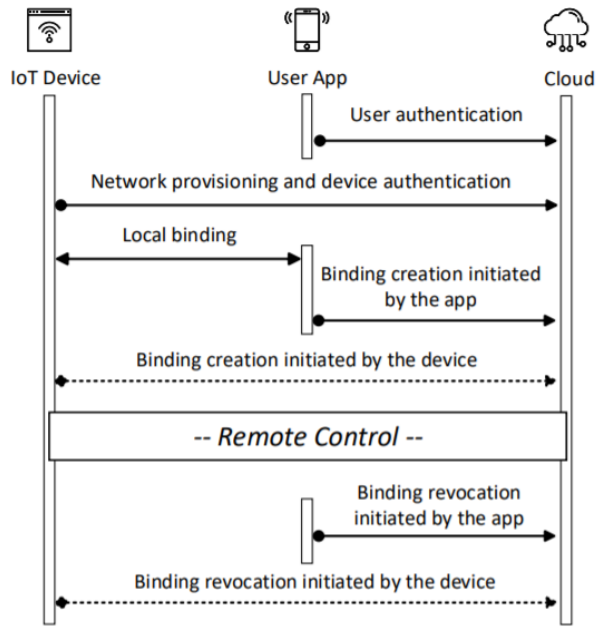

Fig. 5. Procedure of remote binding[20n]

As we mentioned above, the SSID and the Access point's password were encoded in the length of the packet that was sent on broadcast mode. The smart plug stay on promiscuous mode in order to intercept the packets and decode the password and SSID to connect to Wi-FI.
Once the smart plug is connected to WiFI the process of binding start.
A lot of steps are created and a local binding start. The device will send her information composed by device ID and generated local key to Tuya Cloud6.

The key is used to encrypt communication between the mobile application and the smart plug. That means that the hacker only needs the device ID and its key to communicate with the smart plug. So all we need is to intercept traffic from the mobile phone to get "dev ID" and "Local key" That's why we use Man in The Middle (MITM) attack and we start to intercept traffic from the mobile phone in order to get device ID and the key from the application.

After installing a proxy server, for example "Anyproxy"[20d], we weren't able to intercept any plain text data. This can be explained by the SSL Pinning feature that was added to the application to encrypt the communication between the mobile

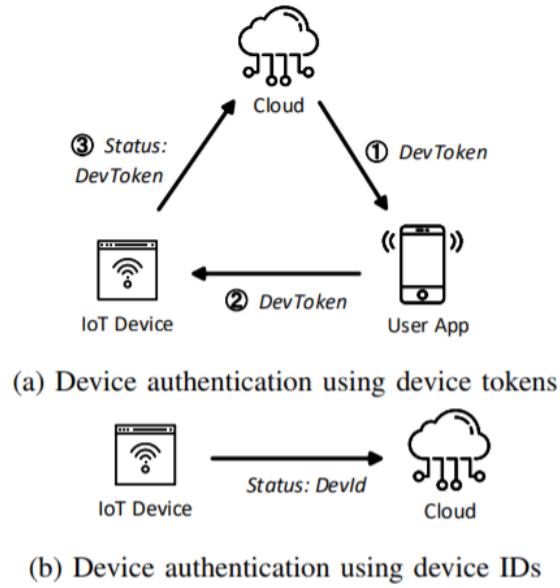

Fig. 6. Device authentication[20n]

application and the cloud. We tried to bypass the SSL Pinning using Frida framework, but we didn't achieve to decrypt the traffic so that's why we switched to the other idea which is using another older version of the application that doesn't have the SSL Pinning feature implemented on it. Since the SSL Pinning feature is configured in the application, the hacker only needs an older application version to intercept the device ID and its key. This is possible with the most older version (v3.4.1)[20j] instead of the latest version (v3.15.1).
After reverse engineering the mobile application, we found that the application communicate on local with the UDP (User Datagram Protocol) protocol and it encrypt the information with the local key that is generated after adding any device to the application knowing that the application work on two modes: the first mode is connecting it to the internet using Tuya Cloud and the second mode is connecting it directly to the device using local area network (LAN) that's why we tried to reverse engineer the method of encryption and by searching on the internet we found that we can use the API created by VTrust that used the same mechanism and the same idea that we discovered when we were reverse engineering the mobile application.
By following the steps from the reference [20a], it's possible to intercept the device ID and its key (see Fig. 7).
The device ID and its key can be used in the Tuyapi project to control the smart plug using API requests using Typescript script[20b] considering that the compromised system can communicate with the smart plug.

IV. Data exfiltration using color variation

A. Problematic

After testing Tuyapi[20b] to read the smart plug configuration, we found that it's possible to change the color mode

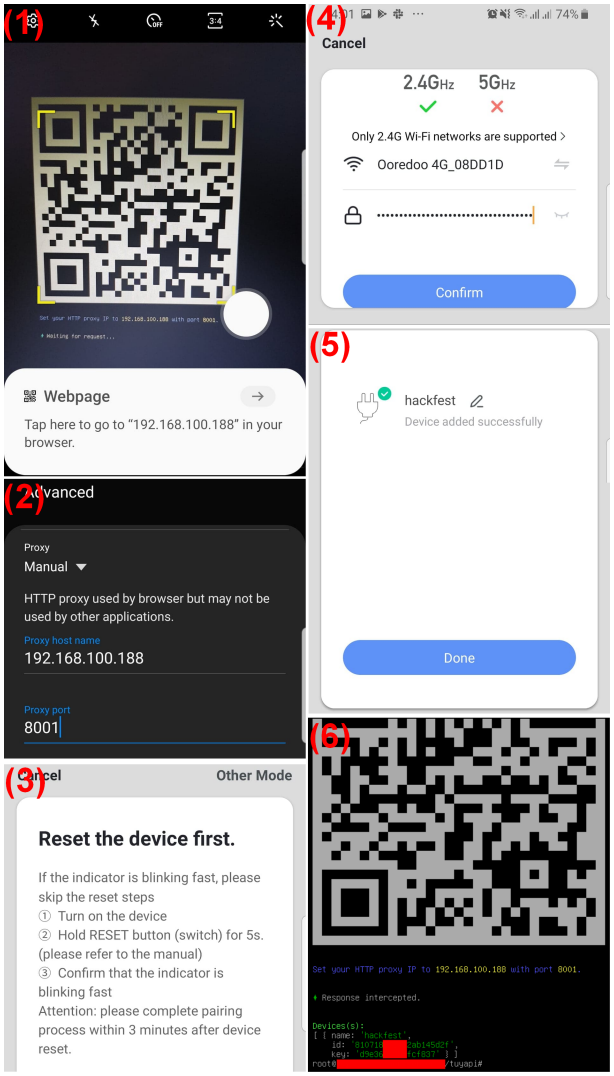

Fig. 7. Device ID and key interception

(white, color and scene) as well as the color of the smart plug's lamp using RGB colors (Red, Green, Blue).
This means, the emitter can change the lamp's color as it wishes. But we have to take in consideration the variables that can lead the data exfiltration to the failure to avoid them from occurring:

- What's the maximum allowed size of the physical receiver device that will be filming the smart plug's lamp ? Installing a big camera in front of people just because it's more professional than the usual smartphone's camera is not going to be a big deal.
- Does filming the plug's lamp require storing the video into the storage disk ? If for some reason the data extraction from the filmed scene fails because the receiving parameters are not correctly set, recording the video and storing it in the storage disk could be useful for an offline data extraction.
- How much time can the receiver device film the video without any interruption ? This includes the environment interruptions (for example the people's movement), the energy breakdown (if the receiver device is not plugged to electricity), the storage disk issues (if the video is recorded into the storage disk and it went full) and the hardware limitations (for some cameras, there are predefined maximum allowed duration time of recording for every video even if that duration time doesn't mean that it will make the storage disk full).
- What's the time interval of detecting two consecutive frames in the receiver's device ? If it's applying some filters to the recorded video this will increase that interval due the heavy processing work. The emitter should send the bits in a time interval that is greater than that mentioned interval otherwise the receiver device will not be able to capture all the exfiltrated bits.

These questions have higher priority than others because ignoring one of them will make the data exfiltration fail.
After that, come the functional questions:

- How to define the color of the bits "0" and "1" whatever the color that was initially configured by the emitter in the smart plug and whatever the defined mode (white, color, scene) ?
- How to detect that the smart plug successfully received the same bit from the emitter (like at least a "0" and then a "0", or a "1" and then a "1") ?
- What to do if the smart plug's lamp is initially set with a color from the gray scale color range when the smart plug is configured with the color mode ?
- How can the data exfiltration be speed up ?
- Can we generalize this mechanism to exfiltrate multiple bits at the same time by modulation ?
- How to deal with the environment brightness (like in the morning, or in the night or in the presence of any other source of light) ?
- How much the physical receiver device is allowed to move during the data exfiltration ?
- How to deal with the imperfection of the camera sensor that doesn't detect a single color from a single place even if the physical receiver device's position is fixed ? This imperfection is intensified more in the darkest environments.
- How to deal with the delay time that is caused by the smart plug when the emitter changes the smart plug's color and it should wait for the smart plug's answer whether the action was successfully performed or not before that the emitter changes again ?
- Is it possible to extract the exfiltrated data from a recorded video offline even if we forget or we don't know the exfiltration parameters that were configured by the emitter when the bits were sent to the smart plug ?

And since, we want to exfiltrate the data by encoding it as bits in a stealthy mode, we also asked ourselves the following questions:

- How to stay under the stealthy mode using the color's

variations that can only be detect by the camera and should be hard to detect by the naked eye ?

- What's the minimum delay of time that should be respected to avoid attracting people's attention with the light blinking ?

B. Color representation definition in the Color mode

The first though about the stealthy mode was to keep in mind that the emitter should not send the data to the smart plug quickly to avoid attracting people attention when the smart plug's lamp blinks quickly. For the color mode, it's important to use the spectrum color range that is hard to detect by the naked eye. Since the eye is less sensitive to the purple range colors (see Fig. 8), it's preferable to operate in this color range.

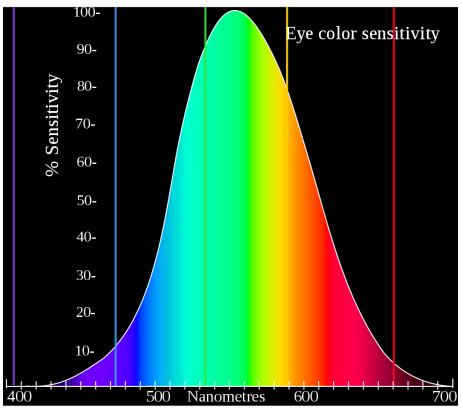

Fig. 8. Photopic relative brightness sensitivity of the human visual system as a function of wavelength (luminosity function)[20f]

But, what makes this solution hard to use is when the initial lamp color is not close to the purple color range. That will open to several questions like is it acceptable to change the lamp's color to a suitable color at least temporarily in beginning of the operation in order to detect the color change in a stealthy mode until the end of the operation ? If moving to the mentioned color range is not allowed, the emitter should set the lamp's color with a color closer than the actual configured color.

But, the RGB color representation is hard to use in order change the color to another closer color (see Fig. 9) because in RGB color representation we need to change the color intensity of the three colors (red, green and blue).

Fig. 9. RGB color representation[20g]

After reading about the other color representations[20g], we think that the best color representation that fit with our need is most likely the HSV (Hue, Saturation, Value) color representation (see Fig. 10). This color representation mode was also adopted by the mobile application "Smart Life - Smart Living" (see Fig. 3).

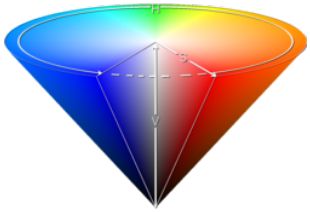

Fig. 10. HSV color representation[20g]

The H (Hue) component controls the color variation in a circular way, which can be used to define the bit values ("0" or "1"). The S (Saturation) component, that have an horizontal variation (see Fig. 10), controls the brilliance and intensity of a color, which can be used to inform the receiver that there is another bit that was successfully sent to the smart plug. The V (Value or Brightness), that have a vertical variation (see Fig. 10), controls the lightness of the color will not be used here as a personal choice to make the study simple. But to generalize this study, the V component can be used to send extra bits or it can be correlated with the S component to inform the receiver that another bit/parameter was changed.

Choosing the S component rather than the V component to inform the receiver that a bit was changed is a personal choice that was made during this study.

But whatever the component that we opted to use to do this, will come with a big problem for the receiver which is the center of the conical shape of the Fig. 10 which defines the gray scale color range. Working near this color range is problematic because of the color variation will be hard to detect since the presence of the pure color is very low due to the influence of the low value of the Saturation (or the Brightness). In this case, the emitter have to change the lamp's color in order to step away from the center of the conical shape. This will allow the receiver to detect the color's variation.

C. Color representation definition using one HSV's component

It's possible to exfiltrate data by sending bits only using a single HSV's component. For example in the White mode, the only component that we can work with is the Brightness since the lamp's color will be in the gray scale color range (from black to white).

Also using a single component instead of two or three components can increase the level of stealthiness by minimizing the number of variables that we work with. In the next sections we will be demonstrating how this can be possible.

V. Encoding techniques used for the data exfiltration

To cover some of the problematic's questions, we used two different encoding techniques for the data exfiltration.

A. Manchester code technique

”Manchester encoding is a synchronous clock encoding technique used by the physical layer of the Open System Interconnection [OSI] to encode the clock and data of a synchronous bit stream.”[20h]
One of the advantages of Manchester encoding is that it requires only one variable to encode all the data. In our study it’s about one of the HSV’s components. It could be the Brightness (V) from the White mode or any component from the Color mode.

The clock from the Fig. 11 defines the interval of time between two transmitted bits defined by the emitter and that should be known by the receiver in order to successfully decode the encoded data. Any issues that come along with a wrong clock rate will make the decoding impossible. The data from the Fig. 11 defines the sequence of bits intended to be encoded using Manchester code then sent to the smart plug and that should be received by the device receiver and decoded there. And the two Manchester representations from Fig. 11 defines two ways of data encoding that should be respected by the emitter and the receiver.

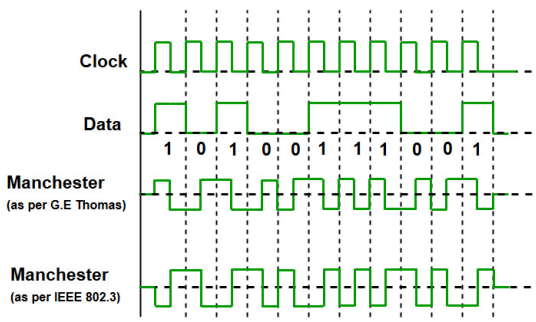

Fig. 11. An example of Manchester encoding showing both conventions for representation of data

Using this technique we can ensure the usage of a single variable to make the operation more stealthy.

B. Circular technique

The name of this technique come from the usage of the variation of the Hue to modulate the data. As we previously mentioned in a previous section, the Saturation can be used as detect the transmission of new bits. This means that the Circular technique can only be used in the Color mode.
This technique is useful to send multiple bits at the same time by modulating a sequence of bits into multiple colors. The number of colors that can be defined by the emitter to be sent can be calculated as follow:

$$T = 2^{(n+1)}$$

Where T is the total number of colors that are shown in the smart plug, n is the number of bits that will be used in the data modulation. The $+1$ in the exponent comes from the fact that for every value of the Hue, there are two associated values of the Saturation.

For example, if the emitter is sending two bits once, it needs to modulate the bits two by two. There is going to be 8 values of colors calculated as follow:

$$T = 2^{(2+1)} = 2^3 = 8$$

But from the Hue’s perspective it’s only 4 values. The additional values of colors are only needed by the emitter to identify the bits that are consecutively being transmitted with the same value. This means that the emitter should always change the saturation value for every transmitted bit. And this only require at least 2 values of saturation that will be used for all the values of the Hue. There is no need for additional values of the Saturation.
When the receiver detect that the saturation changed, it should be aware that a new bit was transmitted. Then, it should be able to identify the detected color refers to which sequence of bits. That’s how the process should be done.
The number of transmitted bits at the same time defines also the level of stealthiness. The less the number of bits are used at the same time, the more the operation is going to be stealthy but sometimes due to some reasons, threat actors prefer to favor the rapidity of the operation when the conditions allows that rather than waiting a long time during the operation.

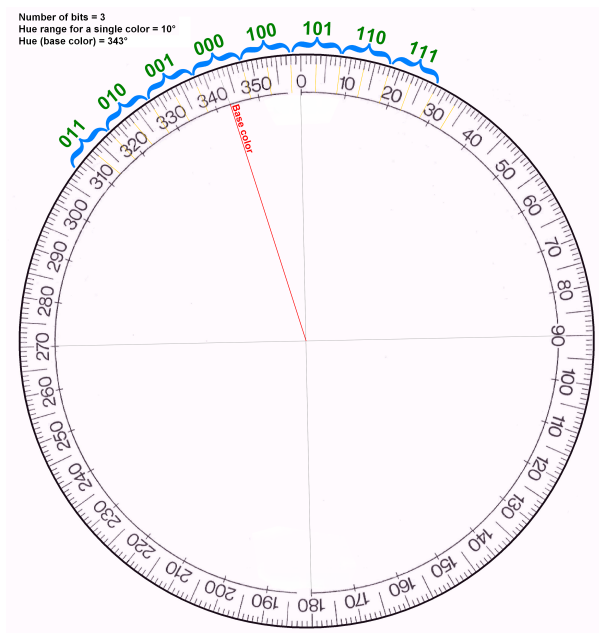

Fig. 12. Bits modulation using the Hue circle

The Fig. 12 demonstrates how the circular technique allows the modulation of the data, using the Hue circle, over multiple colors. In this example, we used 3 bits. We supposed that we will be working around the pink color range. The first value of the Hue should be a reference since the other seven values of the Hue are going to be around the first value that we called ”Base color”. In this example, the base color had a Hue’s

coordinate with 343 degrees. In the same figure we defined the Hue range for a single color as 10 degrees. This means that if the receiver detect any Hue's value within any of the defined Hue's ranges, that will value will be associated. In the same example, if the detected Hue's value is 11 degrees, that means that the sequence "110" was transmitted to the smart plug.

VI. Receiver's detection techniques in practical

A. Introduction

During this project, we tested the implementation of these techniques in an Android mobile application using Java for a real time data extraction and in a Python 3 script for an offline data extraction and both results were the same. But in this study, we found that theory is too perfect to be easily applied in practical without any tweaking.

B. Color detection

The first thing that needed to be discussed in the beginning is what was mentioned in the problematic which is how to deal with the the imperfection of the camera sensor that doesn't detect a single color. After trying an experiment about showing the HSV values of the colors that were detected by the receiver's camera (with a fixed position) in a single pixel took from the center of every frame with, we found the following results:

- In the daylight, the values of the HSV components are changing a little bit but these changes are high to tell exactly the real HSV's values.
- In the nightlight, the values of the HSV components are changing more that we can't even tell what's the correct color that the camera is detecting.

If a single pixel can't help to identify the real HSV's value of the detected color, then a range of pixels around a single point with a specific radius can give an approximated value of the HSV when we calculate the average of every HSV component separately. And this was an efficient solution after performing the same previous experiment.

C. Distance between two Hue's values

The Hue values goes from 0° to 360° in a circular representation (the 360° is converted to 0°. Calculating the distance between two Hue's values is an indispensable thing that should be done correctly where $distance \in [0, 360[$. One of the correct ways to get the distance value can be as follow:

$$D = |newColor - oldColor|$$
$$if\ (D < 180)\ then$$
$$R = D$$
$$else\ if\ (D > 360)\ and\ (oldColor < newColor)\ then$$
$$R = |(oldColor - (newColor - 360))|\ \%\ 360$$
$$else\ if\ (D > 360)\ and\ (oldColor > newColor)\ then$$
$$R = |((newColor - 360) - oldColor)|\ \%\ 360$$

Where D is the difference between the new color and the old color and R is the real shortest distance between the Hue's values.
As an example:

- If both Hue's values were detected as 13°, the real distance will be set as $13 - 13 = 0$ according to the first condition.
- If $oldColor = 13$ and $newColor = 5$, the real distance will be set as $13 - 5 = 8$ according to the first condition.
- If $oldColor = 347$ and $newColor = 5$, the real distance will be set as $|5 - (347 - 360)|\ \%\ 360 = |5 - (-13)|\ \%\ 360 = |5 + 13|\ \%\ 360 = 23$ according to the second condition.
- If $oldColor = 5$ and $newColor = 347$, the real distance will be set as $|(347 - 360) - 5|\ \%\ 360 = |(-13) - 5|\ \%\ 360 = (13 + 5)\ \%\ 360 = 23$ according to the third condition.

D. Associating the detected colors to bits

1) Problems related to the association between the detected colors and the transmitted bits: During all the experiments we went through a big problem that prevented us from finding the same HSV component's values as those sent from the emitter. This issue came from the following sources:

- The environment's lightning since the light applied to an object will change its darkness and brightness. In most of the cases that will affect the color's saturation and knowing that if the HSV's component's values is altered, it's impossible to find the configured color that was set by the emitter in the smart plug.
- The color shown by the smart plug that tend to be close to the configured color but it's not really accurate that can be noticed with the naked eye. For example if we configure the smart plug with a Brightness equal to zero, that will not turn off the smart plug's lamp since the shown color should be black but that will still show the configured color, according to the HS components, with a tiny brightness.
- The receiver's camera's hardware issues due to the imperfections of the device that is way far from detecting the correct colors as the naked eye. If the target of the camera's viewfinder is the source of light, that will create bright zones mainly in the center of the source of light that goes white and dark zones far away from the source of light. The Fig. 13 illustrate this issue when the camera is recording a video focused on the smart plug that is configured with a pure red color with 100% of saturation and brightness. The center of the source of light turns to be white instead of being red.

2) Real detected values of the Hue and Saturation components when the Hue is changing: In the following experience, we wanted to see if we fix the Saturation and the Brightness values to 100% in the smart plug's lamp and we keep changing the Hue's value with a fixed step, what are the respective colors that the receiver's device is going to detect.

Fig. 13. Captured frame from a video record focused on a smart plug that is configured with a pure red color

Theoretically, the colors should be pure since the Saturation and the Brightness are fixed to 100%. And the receiver should detect an approximation of these fixed values.
During this experience, the emitter sent to the smart plug's lamp a new configuration to set a new color every 4 seconds. Every new color have a Hue incremented by 5 degrees until turning around the Hue's circle (72 time). The initial color was a pure red color ($Hue = 0$). This experience was performed

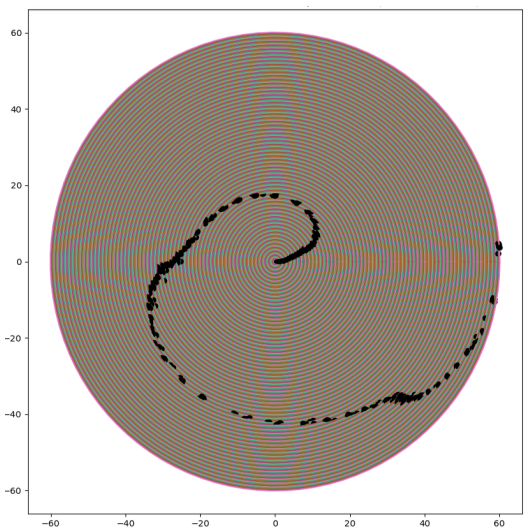

Fig. 14. Hue's variation over the time in a circular representation from a developed mobile application

using a developed mobile application that integrated filtering features.
We extracted the Hue's variation (see Fig. 14) and the Saturation's variation (see Fig. 15) over the time.
The theoretical Hue's figure should be a spiral shaped for every group of 8 circles since the color should changed every 4 seconds and the distance between two processed frames from the receiver's side was defined as 0.5 seconds which makes $4 * 2 = 8$ circles that represents 8 detected Hue's values. But the shape is not perfectly spiral shaped.
Also the theoretical Saturation's value should be around 1 (as 100%). But looking at the Fig. 15, will show that the Saturation changed depending on the Hue's value.
In practical, we experienced another thing that is even if the receiver's camera is calculating the average color of a specific pixel's range, the color changes over the time also due to the imperfections of the receiver's camera that can't detect the

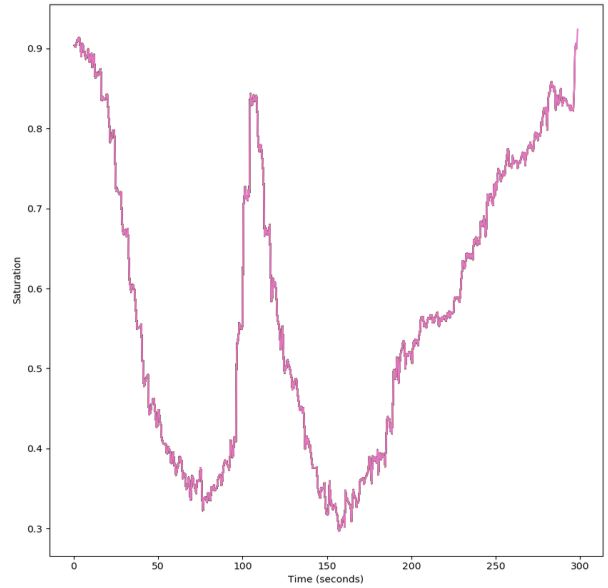

Fig. 15. Saturation's variation over the time from a developed mobile application

color that was instantly changed in the smart plug's lamp but instead it will detect the transition colors, in a non-linear or random way, until that the color will stabilize around the seen color. We did the same previous experience using the native mobile camera without applying any filter and with disabling the auto focus feature leaded us to the results in the Fig. 16 and the Fig. 17 that shows the Saturation's values that doesn't look smooth compared to the results for from the developer mobile application. Also, after that the Hue increases to a specific value caused by the $+5$ of Hue's increment, sometimes it increases with a variation less than 5° and sometimes it increases with a variation greater than 5°. The Fig. 18 shows

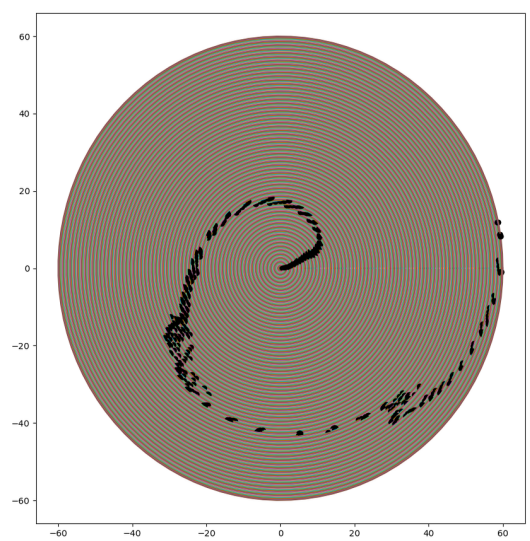

Fig. 16. Hue's variation over the time in a circular representation from the native mobile camera

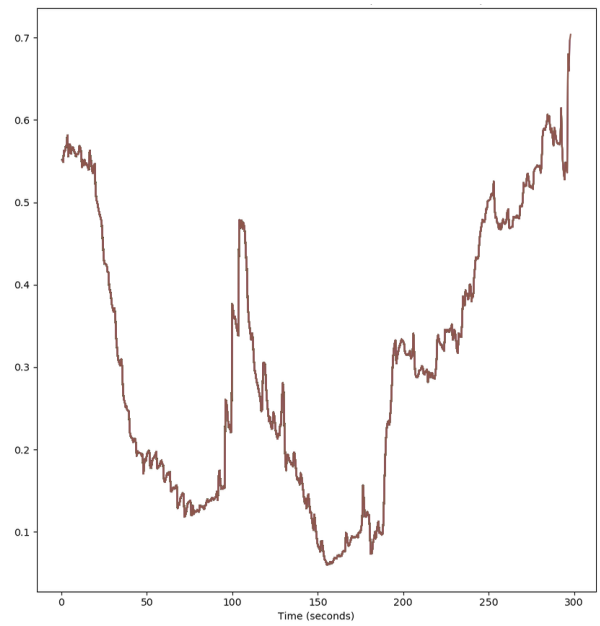

Fig. 17. Saturation's variation over the time from the native mobile camera

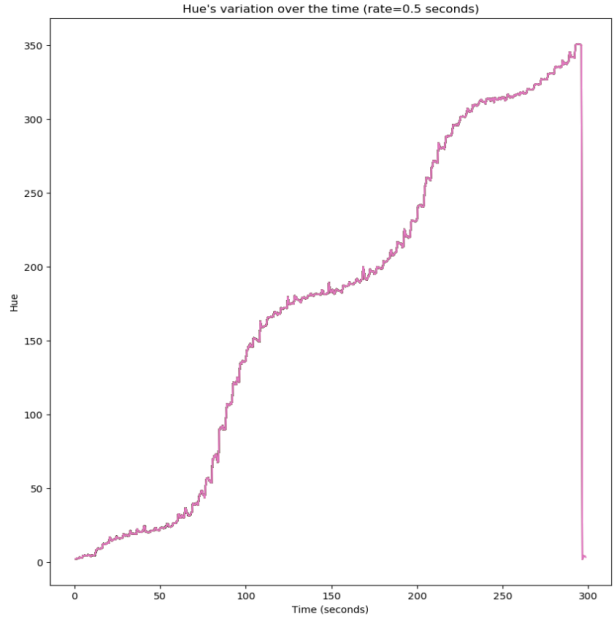

Fig. 19. Hue's variation over the time from the developed mobile application

better this variation that can reach $+25$ when we are not applying the necessary filters but that is smoother when these filters are applied (see the Fig. 19).

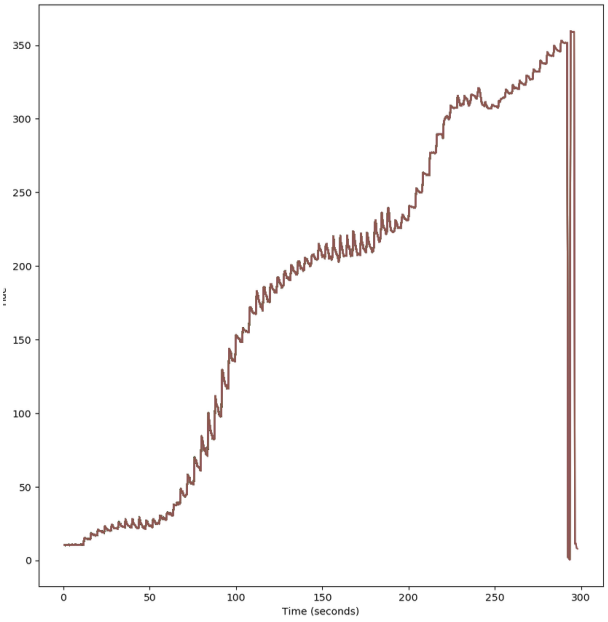

Fig. 18. Hue's variation over the time from the native mobile camera

3) Accurate association between the detected color and the transmitted bits: Knowing that using the color range distribution over the bits blindly is theoretically interesting but practically it's not always working since changing any of the HSV components could be detected from the receiver's perspective as another component that was changed. This is why we though that if we know each color represents which bits, instead of distributing the color ranges over the bits we can simply associate the HSV values to the value of the bits. The idea was to add another step before starting the data exfiltration which is the reconnaissance step that will allow the receiver to detect the real colors for every bits values. The receiver will record every color and then it will calculate the distance between every captured color with all the colors from the recorded colors. The reconnaissance step will end when the recorded colors list includes at least two colors and the first recorded color is detected again. The next detected color will mark the beginning of the data exfiltration and the detected colors should be close to one of the already recorded colors since the reconnaissance is meant to show to the receiver all the possible colors that it might receive during the data exfiltration. That way it only need to calculated the distance between the current detected color and all the colors that were previously recorded in the reconnaissance. The color from the recorded list that is close to the current detected color will tell what's the detected color and then the real transmitted bit.

4) Color detection improvement: During all the experiences that we performed, whatever the receiver's settings that we set, we wanted to detect the color that the receiver is focusing on. When the area is filled with dark color and the center of the source of light is in white, when we calculate the average value of the HSV color, these colors will add extra noise especially when the color that we want to detect is in a small area like the example of the Fig. 13. This noise will be an issue when the receiver needs to find whether the variations on the HSV components are exceeding the limits or not because of the excessive presence of the black or the white color in an area that contains few amount of another color. In fact, the pure white and black colors are useless when the receiver is working on the variations from the Hue component because these two

colors could not be really locate the color in the circle since these colors are in the middle of the conical shape. This means that excluding the pure black and the pure white colors when the receiver calculates the average value of the HSV of a specific area could improve the color detection. And this is what really happened when we tested it.
Another tip could be useful to improve the color detection is that the area that will be captured should not focused on the white only or the black area only and wait for the receiver to detect high variations in the HSV components. But instead the area that will be captured should be filled with a high amount of the real color that we are seeing with the naked eyes rather than the white or the black colors. This is because the last proposed solution is just a workaround. Excluding the pure black color and the pure white color are not enough because there will be another colors that are close to the pure black and the pure white colors and if the area is filled with these colors there will still be the same noise. And this doesn't mean that the last proposed solution is useless because it's pretty useful especially when the receiver changes the camera's settings (like the sensor sensitivity and the sensor exposure time) for example in order to intensify the presense of the searched color, the white and black areas that are in the same area could turn to pure white and pure black colors and this way excluding them is easy with this feature.

VII. Data exfiltration's workflow

A. General workflow

We have generalized the data exfiltration's workflow in the Fig. 20. This workflow will describe how the data exfiltration was done using the light as a covert channel.

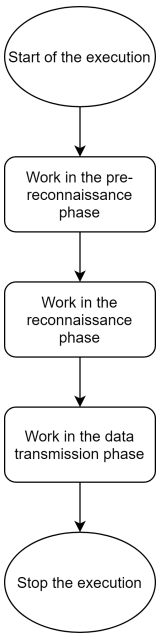

Fig. 20. Data exfiltration's workflow

The pre-reconnaissance step is the phase associated with the initial smart plug's settings. When the emitter and the receiver enters in the reconnaissance mode, the emitter send all the colors that the receiver should be aware of.
The transmission phase is the phase in which the real data will be exfiltrated. The emitter will only use the colors already shown in the smart plug's lamp from the reconnaissance step. The emitter will stop the execution automatically and the receiver could stop it automatically using the automatic stop feature that we explained previously or it could be manually stopped by the user.

B. Emitter's workflow

1) Emitter's general workflow: The emitter's workflow that we defined was described in the Fig. 21.

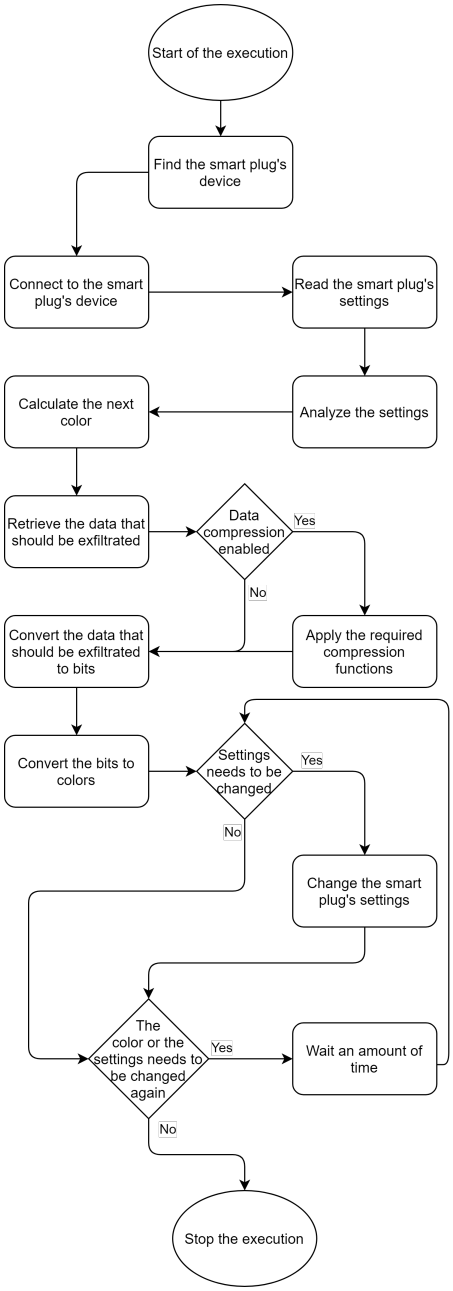

Fig. 21. Emitter's workflow

The conversion of the bits to colors is meant to include the fact that the bits could be modulated to make a specific number of bits being sent at the same time.
Since this workflow could be a generalization used by an emitter during the data exfiltration, we used to mention the settings without mentioning changing the colors since the color is part of the settings that could be set in the smart plug. This means that the settings includes the color value, the color mode and any other settings that could be changed in the smart plug like the smart plug's lamp whether it's on or off.
After changing the smart plug's settings, it's important to wait an amount of time if the settings should be changed again. This amount of time is greater or equal to zero seconds. And allowing the emitter to wait an amount of time could be useful for several reasons like enabling the stealthy mode by setting the smart plug's settings periodically to avoid the smart plug's lamp from blinking quickly whenever the settings are accepted by the smart plug also quickly. This amount of time could be useful also to make the receiver detect the real sent colors that are being sent. We will be talking about how this could be done in the receiver's workflow.
Whatever the number of bits that are going to be transmitted at once, the only thing that will be considered is the colors that will be shown by the smart plug's lamp since the receiver should detect the lamp's colors one by one.
The mentioned need of changing the smart plug's settings could be caused by the fact that the color should get back to its initial value after finishing the data exfiltration to remove all the traces of any suspicious alteration in the smart plug's settings. This is why if there is no need to change the color again, the execution will stop.

2) Emitter's workflow with the adopted encoding techniques: We created two Javascript scripts that implemented the emitter's workflow. Each script was intended to enforce an encoding technique (Manchester encoding and the circular encoding). Both scripts requires the text that will be exfiltrated over the smart plug, the number of the transmitted bits (it's fixed to one for the Manchester encoding technique), the variation's value of the Saturation. And in addition to that, the script that is enforcing the circular encoding technique requires the variation's value of the Hue.
After finishing the data transmission, the emitter will restore the smart plug's settings as it was from the beginning and the workflow stops.
The Fig. 22 and Fig. 23 shows an example of the beginning of a data exfiltration from the emitter's perspective.

3) Emitter's workflow with the Maxcio smart plug: The created two Javascript scripts were based on Tuyapi[20b] project.
The first thing that we had to identify is how the color mode and the color mode are set. We dumped the settings fields and with the help of "Smart Life - Smart Living" mobile application, we tried to change a single variable (the color mode or the color value) before reading the smart plug's settings again to understand what are the affected fields and how they were tweaked.

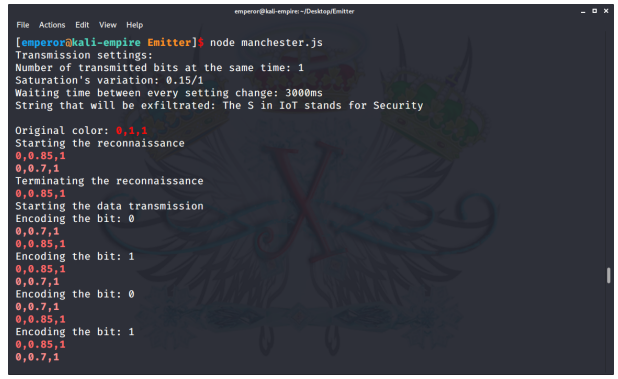

Fig. 22. Example of the emitter's execution using Manchester encoding technique

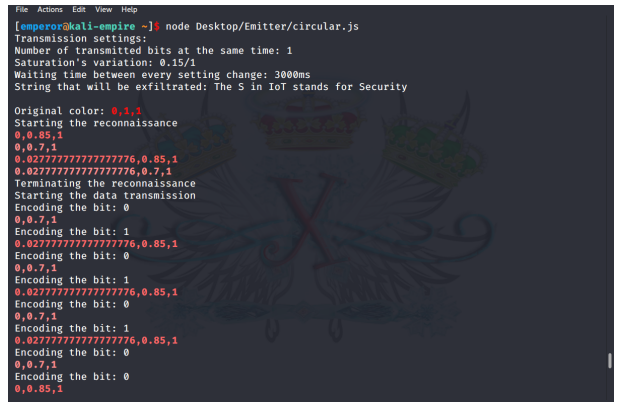

Fig. 23. Example of the emitter's execution using circular encoding technique

In the end of all the experiments we found the following results:

- The white mode is enabled when *dps[28]='white'* and the color value is set as *dps[29]={VALUE}* where {*VALUE*} is the intensity of the white color going from 0 to 255.
- The color mode is enabled when *dps[28]='colour'* and the color value is set as *dps[31]={HEX}ffffffff* where {*HEX*} is the Hexadecimal representation of a color written in 3 bytes. For example, if the color is red (Red=255, Blue=0, Green=0), {*HEX*}=*ff0000*.
- The scene mode with the standard colors (located in one of the modes from the area number (1) in the Fig. 24 is enabled when *dps[28]='scene'* and the color value is set as *dps[32]={HEX}0168ffff*.
- The soft scene mode (see area number (2) from the Fig. 24), that will show smoothly show the color and hide it over the time, is enabled when *dps[28]='scene_1'* and the color value is set as *dps[33]=ffff5001*{*HEX*}.
- The rainbow scene mode (see area number (3) from the Fig. 24), that will show instantly a color among a list of six maximum colors in the defined order and over the time, is enabled when *dps[28]='scene_2'* and the color value is set as

Fig. 24. Scene mode in the "Smart Life - Smart Living" mobile application

dps[34]={HEX1}{HEX2}{HEX3}{HEX4}{HEX5}{HEX6}. The last colors that are omitted or that are set with a white color are not shown by the smart plug.

- The shine scene mode (see area number (4) from the Fig. 24), that will show and hide instantly a single color over the time with a delay of one second, is enabled when *dps[28]='scene_3'* and the color value is set as *dps[35]=ffff5001{HEX}*.
- The gorgeous scene mode (see area number (5) from the Fig. 24), that will show smoothly the colors from a list of six maximum colors one by one in the defined order and over the time, is enabled when *dps[28]='scene_4'* and the color value is set as *dps[36]={HEX1}{HEX2}{HEX3}{HEX4}{HEX5}{HEX6}*. The last colors that are omitted or that are set with a white color are not shown by the smart plug.

If the scene mode have limitations that prevent the success of the data exfiltration operation, then it's possible to temporarly switch to the color or the white mode in order to perform the data exfiltration in the perfect conditions. Also if some other conditions should be took in consideration to make the smart plug's lamp showing the colors as usual, it's possible to change the waiting time and to choose the right colors from the actual smart plug's settings.

C. Receiver's workflow

1) Introduction: We built the mobile application and the Python3 script as receivers with the same workflow. There are some functional differences associated with these 2 receivers that are intended to do the same main thing. We will be discussing in general about the receiver's workflow and then we will mention what's the difference between these two receivers.

2) Receiver's general workflow: The receiver's workflow that we defined was described in the Fig. 25

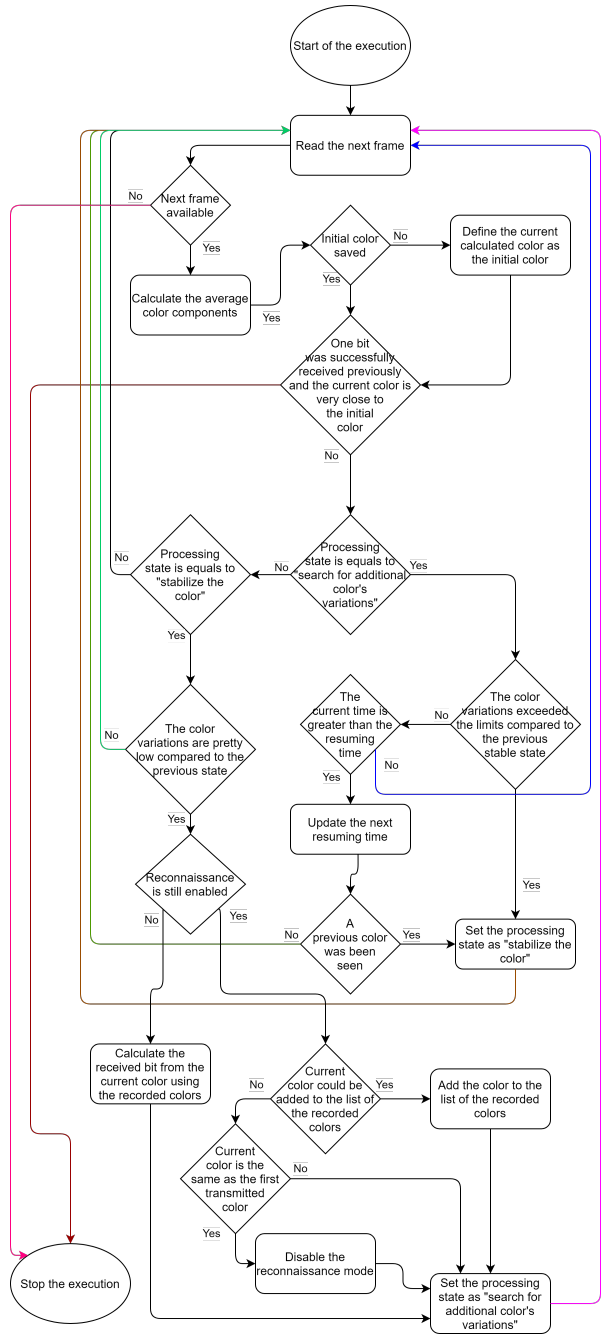

Fig. 25. Receiver's workflow

The receiver's workflow is applied at least just few seconds before the emitter start changing the smart plug's configuration.

While there are video frames that needs to be proceeded, the receiver will continue doing the processing on these frames. The automatic stop feature is a feature that allows the receiver

to stop the workflow when the current detected color is the same as the color that was initially detected before starting the reconnaissance step which also informs the receiver that the smart plug's settings were restored and the emitter's workflow is done.

The processing state is a variable that can tell what's the current receiver's objective. Initially, the receiver have to detect the high variations in the calculated color. If these variations exceeds the defined limits, the receiver will know that probably the emitter successfully changed the smart plug's settings and the receiver have to make sure whether this possibility was correct or not by changing the processing state to "stabilize the color" which allows the receiver the start the verification whenever the color is stable and it's not changing too much according to the variations limits. And whenever the color is stabilized, the receiver check in which step it was working on. If the actual state is the Reconnaissance step, which is the first step in the data exfiltration, then it should build a list of unique colors detected from the smart plug as colors that are not close to each others since during the Reconnaissance step, the emitter shows to the receiver all the color values that it might aware of and they are associated to which bits so that the receiver will be able to deal with the next step which is the data transmission. Now, getting back to the main condition, if the actual step is not the Reconnaissance step, it should be the data transmission step in which the receiver have to search for the closest color to the current color from the recorded colors obtained from the reconnaissance step. This will allow the receiver to know the bit associated to the actual color. And whatever the results got when the processing state is equals to "stabilize the color", the receiver should search again for another the high variations if data exfiltration is still running and there are other frames that should be processed.

As an important note, when the reconnaissance is done, the receiver finish the calculation of the time frequency's value which is the time distance between two colors changes in the smart plug. For Manchester code technique, this time will be used as reference to synchronize the timer every time the color changes really. Technically, every time a new color is confirmed to be changed and it's stable, the receiver set the next time limit as the current time plus the time frequency multiplied by 1.25. And whenever this time is exceeded by the current time without seeing any color changes, the receiver knows that the emitter didn't changed the color because it was sending the same Manchester bit twice (when a real bit is changing from 1 to 0 or from 0 to 1, the second Manchester bit associated to the first real bit and the first Manchester but associated to the second real bit will be identical). And in this example if we suppose that the color changes every 4 seconds, if the color didn't change within 5 seconds, the receiver will consider that the same Manchester bit was transmitted. And of course there will not be a third same transmitted Manchester bit. This is why we think it's acceptable to use a the 0.25 of delay to know whether the same Manchester bit was transmitted or not. And since the next color will change for sure, the receiver could synchronize its clock again and set the next synchronization time again as we have previously described. For the circular technique this feature is not needed because the color will change for sure since the Saturation will act as the triggered for these changes.

Going to the details, one of the most hard things to identify in this workflow is the actual color should be compared with which color to make the mentioned assumptions that there was a high variation in the color or that the color was stabilized. We went through a lot of issues every time we make a decision about the other color that should be compared with the current color.

During this project, we chosen the previous color as a reference. But we found that the previous color couldn't be always the correct value that should be used because of the following issues:

- Sometimes the real time detection goes wrong when we tested the workflow and sometimes when a new color is shown by the smart plug's lamp, the receiver detects the color shown by the lamp but it couldn't tell that there was high variations in the detected color because the camera wasn't be able to detect the changed color instantly but instead it detected it progressively after capturing additional transitional colors in every frame until detecting the real changed color. But looking at the variation between every frame will show that there was not high variations between the transitional colors but there was high variation if we consider using the previous stable color.
- After using the offline receiver by providing him the video recording of the data exfiltration scene, when the time distance between two processed frames is low than usual, the reconnaissance step doesn't finish because the variation limits are too high and decreasing the limits artificially when the time distance between two processed frames is low will not be useful because it's true that time distance should not be high but it could be low to work on a great number of frames to get accurate results.

The issue now got attributed to the high variations only. So if we can work with the previous stable color instead of the previous color shortly to check for the high color variations, we could improve the detection. And as mentioned in the diagram, the high variations were got from comparing the current color with the stable color unlike the stable color state was confirmed from comparing the current color with the previous color.

3) Synchronous receiver: The synchronous receiver is meant to extract the exfiltrated data in real time while the color is changing in the smart plug's lamp.

The developed mobile application was the synchronous receiver. It uses the device's camera to read the scene frames in real time and it's extracting the exfiltrated data from it while it's recording it.

The synchronous receiver doesn't need to move to the next calculated frame to extract the data from it since the next frame is always available when currently because the time is

moving and there is no need to increment the time to get the next frame in a real time video recording.

4) Asynchronous receiver: The asynchronous receiver is meant to extract the exfiltrated data anytime using the recorded video that was taken using a camera.

The Python3 script is an asynchronous receiver since it only needs a video file as an input to extract from it the exfiltrated data.

The asynchronous receiver always need to extract the next frame using time calculation since the video is already recorded so it should be parsed. One of the advantages of the asynchronous receiver is the fact that it could extract statistical information from the recorded video that helps the user to calculate :

- The time distance between two processed frames.
- The limit value of the Hue's variation.
- The limit value of the Saturation's variation.

D. Testing

1) Overview: We used a mobile application as a receiver. The mobile application was developed using Java in an Android device that is supported in our smartphone (API>21).

The mobile application is configurable to customize the sensor sensitivity (ISO), the sensor exposure time, the zoom, the radius, the hue's variation's limit, the saturation's variation's limit and the decoding technique (Manchester code mode or circular mode) based on the central pixel on the screen (see Fig. 27 as an example when the application was still recording the scene and see also the Fig. 28 as a similar example from another recorded video with the same smart plug's settings but this figure was showing the mobile application when the data exfiltration was done and when the mobile application finished recording the scene).

In the mentioned figures, the mode could be changed by clicking on its value and choosing any of the two modes from the dropdown menu (see Fig. 26).

We made the mobile application save any mobile application's changes locally to prevent loosing them and to be able to resume the work any time.

Moreover and as a single detail, we created the central arrow with six colors (black, white, red, blue, green) to distinguish the central point in any zone. We created the zoom's limits using the same approach as well.

After configuring the application and before running the emitter script, the mobile application should start recording to capture the initial color that will be used as a reference whenever the data exfiltration is done.

2) Manchester code technique: We tested several examples using the Manchester code technique:

- An example where the lamp's initial color is set as red (with the hexadecimal value *ff0000*).
- An example where the lamp's initial color is set as purple (with the hexadecimal value *8732e1*) (see Fig. 28) in order to test the stealthy mode.

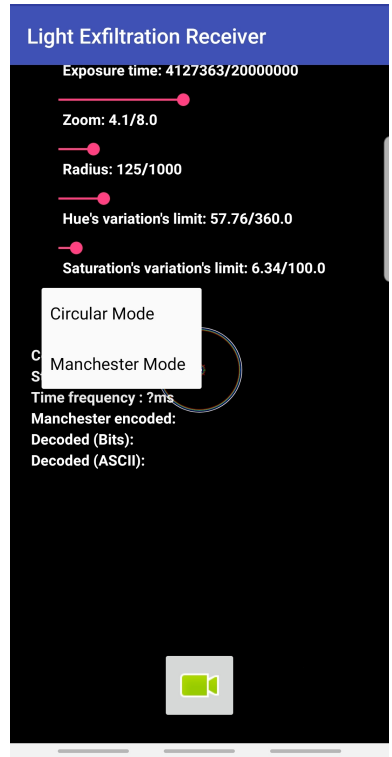

Fig. 26. Example of the end of the data exfiltration using a purple color took from another recorded video with the help of the mobile application

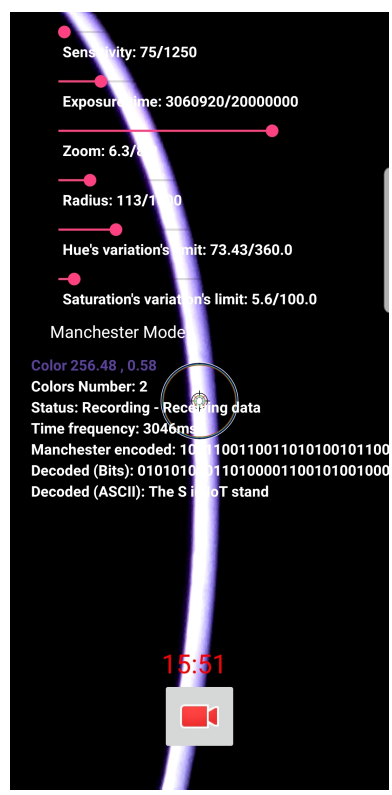

Fig. 27. Example of a real time data exfiltration using a purple color while the mobile application is still extracting the exfiltrated data

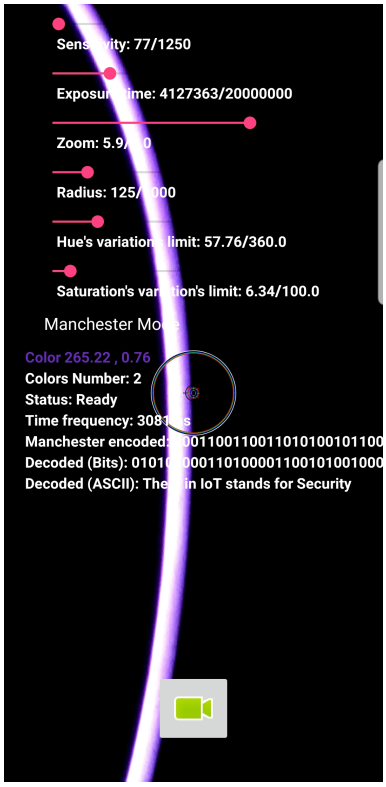

Fig. 28. Example of the end of the data exfiltration using a purple color took from another recorded video with the help of the mobile application

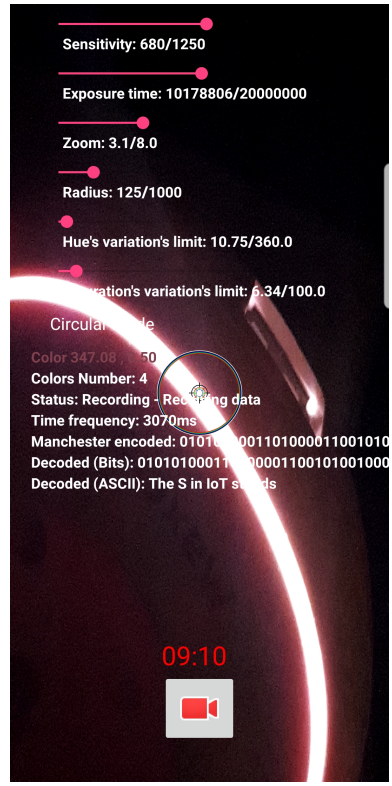

Fig. 29. Example of a real time data exfiltration using a red color while the mobile application is still extracting the exfiltrated data

For all these examples, the emitter changed the Saturation with a 15% of variation every 3 seconds (see Fig. 22) since our mobile phone fails sometimes to extract the data when we decrease these values. This is because the mobile application is doing a heavy work to calculate the average color, depending on the configured radius, and to add some filters on the shown video preview.
The data exfiltration ended within 25 minutes after exfiltrating the sentence ”The S in IoT stands for Security”.
The first example was also tested with a professional camera instead of a mobile phone to check whether the asynchronous receiver (the Python3 script in our example) could extract the exfiltrated data from the recorded video.
And all these experiences were successful even after using the purple color that was very hard to detect with the naked eyes but that wasn't an issue for the mobile phone camera.

3) Circular technique: We tested an example using the circular code technique which involved setting the lamp's initial color as red (with the hexadecimal value *ff0000*) (see Fig. 29 when the data exfiltration is still ongoing and see Fig. 30 when the data exfiltration is done).

The emitter also changed the Saturation with a 15% of variation every 3 seconds (see Fig. 23) with the same reason as described before.
The data exfiltration ended within 15 minutes after exfiltrating the sentence ”The S in IoT stands for Security”.
And all these experiences were successful for some colors.

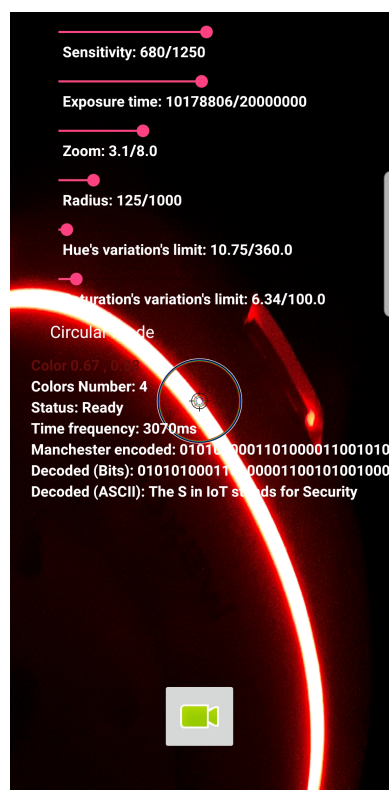

Fig. 30. Example of a real time data exfiltration using a red color while the mobile application finished extracting the exfiltrated data

VIII. Conclusion

This paper reports a study about the data exfiltration using light and particularly we experienced it using Maxico Smart plug. We went though how we could exfiltrate data throw covert channel using IoT devices. Actually, we have successfully created the emitter, the synchronous receive and the asynchronous receiver that works as expected. The Manchester code technique worked perfectly and the circular technique worked for the tested red color and probably for other colors but it was expected to have its issues from the moment when we did the experiment where we fixed the values of the Saturation and the Brightness components and we changed the Hue's values 360° to check whether the receiver could detect the same HSV values or not. And since the results proved that it's possible that for some camera, the Hue's variation from the emitter could imply for the receiver that the Saturation changed and this could be also applied to any tested HSV component. And we believe that this could be the cause of the circular technique that is not perfectly working for some specific colors (or that could work when we count on the asynchronous receiver after delicately defining the limits of the Hue and the Saturation), probably because the calculation of the distance between the colors was based on the Hue and the Saturation components instead of taking in consideration the Brightness component. But this should not a failure for us since even though the results of the experiments on the circular technique were not perfect, since we retrieved from it useful information, it should help the researcher to know what were the problems around it and how to improve the solution. And even though we made a generalization for the data exfiltration using the light theoretically, practically it required a lot of adaptations with additional conditions to take in consideration all the issues described in the problematic. And we have successfully shown how to successfully exfiltrate data using at least one confirmed technique which is based on the Manchester code technique and the other technique (circular) could be improved to make it work properly not just for few color ranges.

We believe that the mobile application could be improved to work also as an asynchronous receiver in order to extract the data from an already recorded video.

The emitter workflow included the data compression functions and unfortunately that was not really implemented but that should be a great idea to reduce the duration of the operation especially. This requires additional design efforts to inform the synchronous receiver whether the compression functions were enabled and which function was used especially that we believe there are two types of useful functions: those used for the big transmitted data and those used for the small transmitted data.

The only limitation related to this study is that the hacker needs an insider threat in the company to intercept the device ID and its key.

The implemented data exfiltration mechanism can also be used in other systems that supports colors variation since most of the study was dealing with the subject in a general way before showing the results particularly with the smart plug.

References

[19] "Covert Channel, data exfiltration". In: *Sothis* (March 14, 2019). DOI: https://www.sothis.tech/en/covert-channel-data-exfiltration/.

[13] "CC3000 Smart Config - transmitting SSID and keyphrase". In: *Depletion Region* (October 7, 2013). DOI: http://depletionregion.blogspot.com/2013/10/cc3000-smart-config-transmitting-ssid.html.

[20a] "Linking a Tuya Device with MITM". In: *Github* (February 15, 2020). DOI: https://github.com/codetheweb/tuyapi/blob/9b53b03baae137d3d2e5b3c4436e384aaeca5b7b/docs/SETUP.md#linking-a-tuya-device-with-mitm-deprecated.

[20b] "Tuyapi". In: *Github* (February 15, 2020). DOI: https://github.com/codetheweb/tuyapi/tree/9b53b03baae137d3d2e5b3c4436e384aaeca5b7b/.

[20c] "'INTERNET OF THINGS' CONNECTED DEVICES TO ALMOST TRIPLE TO OVER 38 BILLION UNITS BY 2020". In: *Juniper Research* (2020). DOI: https://www.juniperresearch.com/press/press-releases/iot-connected-devices-to-triple-to-38-bn-by-2020.

[20d] "Anyproxy". In: *Anyproxy* (2020). DOI: https://anyproxy.io/en/#install.

[20e] "Basic Smart Config". In: *Read the Docs* (2020). DOI: https://sming.readthedocs.io/en/latest/_inc/samples/Basic_SmartConfig/index.html.

[20f] "Color Vision". In: *Wikipedia* (2020). DOI: https://en.wikipedia.org/wiki/Color_vision.

[20g] "Colorizer". In: *Colorizer* (2020). DOI: http://colorizer.org/.

[20h] "Manchester Encoding in Computer Network". In: *Geeks For Geeks* (2020). DOI: https://www.geeksforgeeks.org/manchester-encoding-in-computer-network/.

[20i] "Smart Life". In: *Google Play* (2020). DOI: https://play.google.com/store/apps/details?id=com.tuya.smartlife&hl=en.

[20j] "Smart Life versions". In: *APKPure* (2020). DOI: https://apkpure.com/fr/smart-life-smart-living/com.tuya.smartlife/versions.

[20k] "The Cyber Kill Chain". In: *Wikipedia* (2020). DOI: https://en.wikipedia.org/wiki/Kill_chain#The_Cyber_Kill_Chain.

[20l] "Tuya smart FAQ". In: *Tuya* (2020). DOI: https://smartapp.tuya.com/tuyasmart/faq.

[20m] "Wireshark". In: *Wireshark* (2020). DOI: https://www.wireshark.org/.

[20n] "Your IoTs Are (Not) Mine: On the Remote Binding Between IoT Devices and Users". In: *Department of Computer Science and Engineering* (2020). DOI: http://yinqian.org/papers/dsn19.pdf.