Extended Functionality Attacks on IoT Devices: The Case of Smart Lights (Invited Paper)

Eyal Ronen, Adi Shamir
Computer Science department
Weizmann Institute of Science
Rehovot, Israel
{*eyal.ronen,adi.shamir*}@*weizmann.ac.il*

Abstract—**In this paper we consider the security aspects of Internet of Things (IoT) devices, which bridge the physical and virtual worlds. We propose a new taxonomy of attacks, which classifies them into four broad categories. The most interesting category (which we call functionality extension attacks) uses the designed functionality of the IoT device to achieve a totally different effect. To demonstrate this type of attack, we consider the case of smart lights (whose original functionality is just to control the color and intensity of the lights in a particular room) and show how to use them to achieve unrelated effects. In the first attack, we use smart lights as a covert LIFI communication system to exfiltrate data from a highly secure (or even fully airgapped) office building. We implemented the attack and were able to read the leaked data from a distance of over 100 meters using only cheap and readily available equipment. In another attack, we showed that an attacker can strobe the lights at a frequency which may trigger seizures in people suffering from photosensitive epilepsy (in the same way that rapidly flashing video games can cause such seizures). In our experiments, we have tested both high-end and lower-end smart light systems, ranging from an expensive Philips HUE system to a cheap system manufactured by LimitlessLED. In addition, we consider other weaknesses of the systems we tested, and propose feasible remedies for the problems we found.**

1. Introduction

The Internet of Things (IoT) is currently one of the hottest buzzwords, and many analysts and consulting firms publish extremely bullish estimates of its potential economic impact. For example, a white paper released by the McKinsey consulting group in June 2015 [1] estimates that within ten years, the total value of IoT devices and services will range between 4 and 11 trillion dollars, which represents up to 11% of the world's economy. With such wildly optimistic forecasts, it is not surprising to find hundreds of papers published about the security aspects of IoT devices, describing a large variety of potential attacks on many concrete types of IoT devices.

While there is currently no consensus about the exact definition of IoT devices, most researchers define them as networked devices which bridge the physical and virtual worlds. IoT devices usually have some particular functionality which was designed to lead from the physical world to the virtual world (e.g., by remotely monitoring the user's heartbeat), or from the virtual world to the physical world (e.g., by controlling the room's temperature via the internet) or going in both directions (e.g., by processing the user's spoken instructions to control his TV). In contrast, a standard PC can carry out arbitrary computations but has no predesigned physical functionality, and thus we usually do not consider it as an IoT device.

In this paper, we would like to introduce a new taxonomy of attacks on IoT devices, which is based on how the attacker deviates from their "official" functionality. We noticed that almost all the attack ideas published so far can be clustered into the following four broad types of attacking behavior:

1) Ignoring the functionality
2) Reducing the functionality
3) Misusing the functionality
4) Extending the functionality

In the first type of attack, the attacker ignores the intended physical functionality of the IoT device, and considers it only as a standard computing device which is connected to the LAN (local area network) and/or to the internet. For example, a hacker may combine many compromised IoT devices into a botnet which can be used to send spam or to mine bitcoins. Alternatively, he may try to penetrate the victim's home network and infect his computers by exploiting the presence of IoT devices on the network, regardless of whether they are a smart electricity meter, or a washing machine which can order additional detergent. In fact, we can expect the victim's IoT devices to be the best attack vectors for hackers since there are going to be many cheap devices made by a variety of small companies with minimal security protections, and it will probably be impossible to upgrade or patch their discovered vulnerabilities.

While attacks of this type can pose a serious security threat, in our opinion they are the least interesting type of attack from a research point of view, since they are applicable to essentially any networked computing device and are not unique to IoT devices. Attacks on networked computers had been studied for decades, and we have a good understanding of both the possible attacks and the possible defenses.

In the second type of attack, the attacker tries to kill or limit the designed functionality of the IoT device: The TV will stop working, the refrigerator will not cool its contents, the lights will not turn on, etc. This can be done by malicious hackers in order to annoy an individual or organization, to inflict financial loss, or to create large scale chaos and panic. In some cases the consequences of lost functionality can be more serious: For example, in internet-connected medical devices such attacks can be fatal. Once again, we are already aware of many such denial-of-service attacks on PC's (where a malware infection can kill the operating system or delete some user files). However, the broader scope of IoT devices opens up interesting new possibilities. In particular, the attacker can use ransomware to temporarily lock an expensive physical device and demand a large payment to restore its functionality. For example, most TV's sold today have smart features which enable them to surf the internet, but they have no firewalls, no antivirus protection, and no monthly security updates. It is quite easy to infect a large screen TV with malware which will display an unremovable splash message demanding the payment of several bitcoins in order to make the TV usable. Alternatively, the hacker can infect a smart refrigerator, and threaten the user that unless he pays, all the food in his freezer will be spoiled within a few hours. Note that such an attack on TV's and refrigerators is much more effective than on home computers since even a cautious user has no way to backup or reinstall a clean version of the device's operating system, and it is much harder to haul large and heavy devices to a repair shop than to bring a PC to a technician.

The third kind of attack uses rather than destroys the designed functionality of the physical device, but does it in an incorrect or an unauthorized way. For example, a climate control device is supposed to cool the house in the summer and to heat it in the winter, but a hacker can reverse this behaviour and cause considerable discomfort. In another example, the hacker can turn on all the lights and open all the faucets as soon as the user leaves home for a long vacation. However, most of these attacks are likely to be irritating pranks rather than serious problems.

By far the most interesting type of attack is the last cluster, in which the attacker extends the designed functionality of the IoT device, and uses it in order to achieve a completely different and unexpected physical effect. This requires more imagination and sophistication, since it is not clear how a hacker can use a smart air conditioner to start a fire, or how he can use an internet-connected Roomba in order to unlock the front door in the victim's house. Such unexpected effects may be popular in TV shows such as MacGyver, but in practice they are not easy to achieve.

In this paper we will explore some unexpected applications of connected LEDs, and use them to demonstrate such functionality extending attacks.

1.1. Smart connected LEDs

Connected LEDs are smart light bulbs that are connected (directly or with the aid of a light controller) to a local LAN, thus allowing the user to control brightness and sometimes color from his computer or smartphone. This enables more advanced features such as connecting the light to an alarm clock, or flickering the room's light when new email is received. Connected LEDs have been a rising trend in the last few years. In 2012 Philips released the connected LED system Hue, which was named ”Product of the year” by Forbes tech magazine [2]. In the same year, LIFX and LimitlessLED made successful kickstarter crowd funding campaigns to build their connected systems. In recent years many big and small players have entered this growing market. This trend had grown from smart home enthusiasts to large scale public systems with Philips providing connected lighting solutions to hospitals [3], factories [4] and even cities [5].

Along with the many benefits and new features, this technology also introduces new risks from the virtual world to our physical lives, and vice versa, since today's connected products are not always designed with security in mind.

For example, we have seen a completely insecure setup process that requires the user to send the password of his WIFI router over the air in an unencrypted from (in the LimitlessLED system we tested); Non existing (LimitlessLED) or unsecured authentication schemes (In the Philips Hue system [6]), that allow any user in the network to control the LEDs. In addition, we have also used in our attacks undocumented and sometimes unintended API options that allow malicious users to misuse and extend the functionality of the connected LEDs.

1.2. Our attacks

1.2.1. Creating a covert channel. In order to protect sensitive data, an organization may want to separate its internal network from the Internet. This might be done by implementing a security gateway with data loss prevention (DLP) capabilities, or even by employing a complete air gaped separation in top secret networks. Sophisticated attackers can try to infect those networks with malware using one time access (either physical or virtual). However, it is much harder to find a reliable and continuous exfiltration channel to leak out sensitive data.

Let us assume that such an organization chooses to implement a smart connected light solution, and connects it to its internal sensitive network. We describe how an attacker can use the connected LEDs to create a covert channel.

The basic idea is to misuse the LED's API to switch back and forth between two light intensities, under the following seemingly contradictory conditions: The two light intensities should be close enough to make the transition imperceptible to the human eye, but robustly distinguishable by a light sensor. Fortunately, this is made possible by the fact that most LED lights adjust their luminosity by rapidly switching between on and off states and adjusting the duty

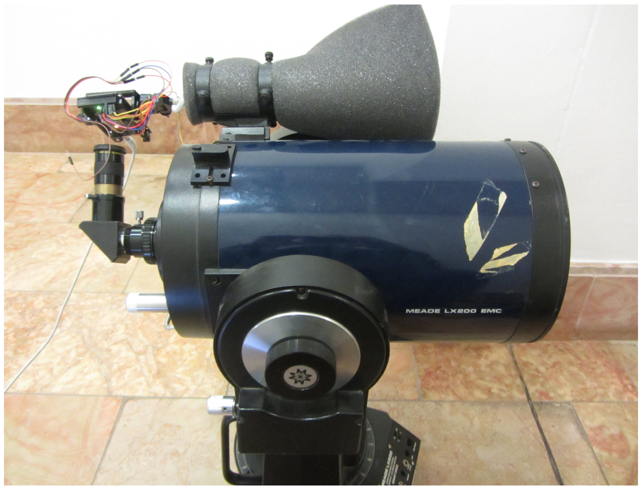


Figure 1. The receiver

cycle. Since the sensor can easily distinguish between such extreme states and accurately measure the duty cycle, it can robustly measure small changes in the light intensity even in the presence of other lights, air turbulence, and other sources of noise.

We built a optical receiver that can accurately read the data from a distance of over 100 meters by measuring the exact duration and frequency of those flickers.

1.2.2. Creating epileptic seizures with strobed light. The risk of epileptic seizures caused by strobed lighting is well known. Our ability to produce crafted signals from the LEDs that we tested, can enable an attacker to create strobes in the most sensitive frequencies. Such an attack could be directed at hospitals, schools and other public buildings using connected LEDs.

2. Experimental Setup

2.1. Overview

Our experimental setup includes two main parts:

1) A transmitting setup that includes a smart LED light bulb, and a controller connected to a PC running our software. Both of them are standard unmodified smart light components.
2) A receiving setup that includes a laptop, light sensor, Arduino board and telescope (Figure 1).

2.2. Building the optical receiver

Our goal was to build a cheap, lightweight and portable setup. As we will see later, our requirements were to detect slight changes in light intensity at interval as small as 10 microseconds, and thus we needed a sampling rate of around 100 KHz. We tried using a smartphone's light sensor, but this option was ruled out as the sensors available on standard smartphones could only measure in the millisecond range. Hence we decided to build a custom device to implement our attacks.

2.2.1. Light sensor. The most commonly used light sensors are photo-resistors or LDRs (light-dependent resistor). The light is measured by reading the voltage divided between the LDR and a serially connected resistor. However this type of sensors has a latency of up to 10 milliseconds which is not fast enough for our requirements.

We chose to use the TAOS TC3200 Color light-to-frequency converter. This sensor converts light intensity to digital frequency output, that for high light intensity can go up to 500 KHz. In addition, It has a very low latency (around 100 nanoseconds). As long as the light source is strong, and our measuring rate is high, we can achieve a very high resolution both in the intensity level and in the time domain.

2.2.2. Arduino. To decode out leak channel we require the ability to measure and collect the light sensor's frequency output at a very high rate and for long periods of time. As this option is only available in high end and expensive scopes, we customized the low level drivers of a 16 MHz CPU Arduino board to allow us to sample the light sensor at a high rate. The Arduino comes with a hardware counter that allows us to count the number of rising or falling edges of the sensor's output. We sample this hardware counter at 10 micro second intervals (100Khz sampling rate), and send it to a laptop (using 160 CPU cycles to measure and send). The post-processing of the data is done on the laptop.

2.2.3. Optics. To decode the leaked data from a long distance we used a 12in Meade LX200 amateur's telescope. The telescope served a dual purpose:

1) Reducing outside light noise by focusing only on the LED.
2) Focusing the flickering light on the small sensor to increase the light intensity measured.

We directed the telescope towards the LED, and mounted the light sensor above the eyepiece as seen in Figure 2. We adjusted the focus of the telescope so that light emitted by the LED was concentrated on the light sensitive part of the sensor.

In Figure 3 we can see a preliminary test of the leak channel in a long corridor from a distance of over 50 meters, with lots of other light sources along the corridor. In later measurements conducted outside the building we achieved ranges of over 100 meters.

3. Attack Description

We tested two connected lighting systems:

1) Philips Lux - a high end white lighting solution by Philips' lighting department. This light system and the color version called Hue are the most

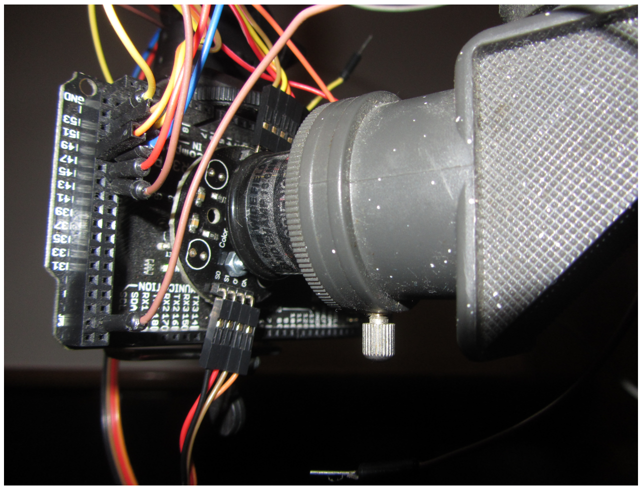


Figure 2. The sensor

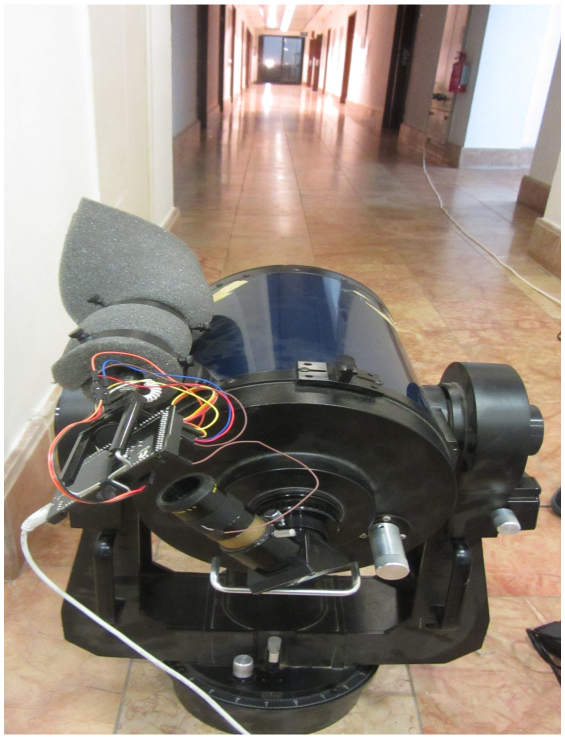


Figure 3. Testing the leak channel

prominent players in the market. We bought a starter kit containing 9 Watt 750 Lumens light bulbs (model LWB004 software version 66012040) and a controller (model PHDL00 software version 1.0).

2) LimitlessLED - A cheap lighting solution that started as a Kickstarter project. It sells the same products under different brand names (we used the one called MiLight). We bought 6 Watt 400 Lumens light bulb with version 3 controller.

We chose those systems as both of them have open APIs which are published and supported by the manufactures [7] [8].

To understand our attack, we need to understand the way the connected LEDs work, the abilities and limitations of their controllers, and how human light perception works.

3.1. Smart connected LED systems

Unlike normal bulbs, smart LED bulbs are designed to be always connected to a power source. They are turned on and off using commands sent by wireless communication from a controlling unit.

In most LED bulbs you can control the power state (on/off), brightness, and in color bulbs also the color of the light. The user can usually operate the light bulb by two main methods

1) A standalone smart switch or remote operated by touch.
2) An application running on a smartphone or computer that connects to a controller using the user's LAN and Internet connections.

3.1.1. The LED bulb. Smart LED bulbs are usually composed of the following parts:

1) RF receiver (and sometimes transmitter) which allows the bulb to communicate with the controller. While some of the smart bulbs (such as LIFX) can connect directly to a network using WIFI or Bluetooth, most of the bulbs use cheaper RF communication and are controlled by a dedicated controller. For example Philips and other companies use a protocol called Light Link [9] over an interface planned for IoT and smart home usage called ZigBee. Those communication channels usually have low bandwidth of around 1 KB/s.
2) Processing unit which is responsible to process and execute the commands received. The processing unit usually controls the LEDs by using PWM (pulse width modulation) signals. The processing unit usually translates the commands into more complex operation. For example, when increasing the brightness, the processing unit increases the brightness incrementally over a short period to avoid sharp light changes that most people find unpleasant (for our attacks we had to find ways to bypass this smoothing feature).
3) Drivers and LEDs. Each bulb usually contains a set of LEDs. In white bulbs the light is generated either from a combination of multiple colored LEDs, or by use of phosphors and one type of LED. The brightness level of the bulb is determined by the signal received by the driver that turns the LED on and off at a very fast rate. The brightness is determined by the ON duty cycle. For example, LimitlessLED has about 27 levels of brightness using a PWM with a frequency of around 3Khz, while Philips Lux has 256 brightness levels using

a PWM with a frequency of around 20Khz. To achieve different colors smart LED bulbs use a combination of monochromatic LEDs, and using the same PWM technique to control each of the LEDs brightness in order to generate a full color range.

3.1.2. The controller. Most of today's systems have a dedicated controller that acts as the gateway between the Internet or LAN and the lights. The controller uses an RF transmitter to send commands to the lights (and sometimes a receiver to receive the light's status). It also has a control interface that is connected to the LAN, and sometimes to the Internet or to a cloud service. Most types of controller can control a number of different lights, and different groups of light, allowing the user to use one controller to separately control the lights in different rooms in the house.

The controller usually enforces restrictions on the rate of commands sent in the system. This might be due to bottleneck issues or as a type of safety method to prevent intended or unintended misuse of the light system.

3.2. Human flicker fusion threshold

The human ability to perceive changes in brightness or color is very complex. Vast empirical research had been conducted in this field. In recent years some of this research was used to design flicker free LED lighting with smooth colors.

One might assume that humans can't detect flickers at a rate above the 24 Hz used in movies and televisions. However, depending on the intensity and color, people can in fact detect flickers at 60 Hz [10], and in some cases of fast movement, a phenomena called phantom array may be perceived even at 200 Hz [11].

To be sure that our covert channel will not be detected we have to flicker at over 60 Hz (over this frequency, the flickering will be fused and seen as reduced brightness). To preserve the same brightness level our flickering has to be done as a PWM signal with very high duty cycle. The threshold is also dependent on the light intensity and the contrast. As reducing the average light intensity is not a valid option (lights are used to illuminate and it will look suspicious if they become dimmer then expected), the other option is to flicker between two close levels of brightness, at the top of the brightness range.

3.3. Leak Method

We looked into several different leak methods. The methods were required to comply with 2 different and somewhat conflicting requirements:

1) Robust - should be resilient to interferences such as other light sources (light bulbs, computer screens, sunlight etc.) and should be easily detectable from a long distance.
2) Not perceivable to the human eye.

3.4. Measuring brightness

Both of the controllers we tested have a recommended rate limit of 10 commands per second, which would normally result in a maximum of 5 Hz flicker frequency. This frequency is well below the flicker fusion threshold, and is very prone to errors from jitters on the network. To address that, we tried to use small oscillations in the brightness level.

Due to the PWM duty cycle method used to control brightness in LED, there are 2 ways to detect the brightness:

1) Average light intensity - measure the averaged intensity over 1 or more PWM's periods. This is easy to measure but is very prone to outside light interferences and has very small contrast between different levels.
2) Measuring the PWM duty cycle - We can detect different brightness levels by accurately measuring the PWM's duty cycles, using the fact that in reality the LED switches between 0% and 100% intensities which are easily distinguishable. This requires a much faster sampling rate, but is very robust to light noise, and there is a very high contrast between the on and off periods.

3.4.1. Measuring brightens changes.

1) LimitlessLED has only 27 brightness levels (which in fact reduce to 24 as the 3 top levels were the same). Unfortunately the changes between adjacent brightness levels were in fact visible to a human observer .
2) Philips Lux has 256 different brightness levels. To provide smooth light Lux works at a very high frequency of 20 KHz or 50 micro seconds. Assuming linear changes in duty cycle we will need to measure differences of around 200 nanoseconds in a period of time between adjacent brightness level. To achieve that we have to use very responsive light sensors and high end measuring equipment.

3.4.2. Creating a crafted PWM signal. To be able to create and measure a reliable covert channel we need the ability to create at least 2 PWM light signals with the following characteristics:

1) Similar duty cycle (to maintain perceivable brightness
2) Off period shorter than about 10 milliseconds (bellow the flicker threshold).
3) Measurable difference in either Period or duty cycle between the signals.

As both lighting systems' published APIs didn't allow us to create such PWM, we had to find undocumented ways.

4. Hacking the LEDs

4.1. LimitlessLED

4.1.1. Security flaws in the setup process. The first security issue was encountered when we tried connecting to the LED's controller. Although the controller uses a standard USB port for power, it does not use this interface for communication. The controller communicates using 2 wireless interfaces:

1) 2.4 GHz RF interface to communicate with the LEDs.
2) WIFI control interface to communicate with smartphones and computers.

To setup the controller as part of the user's WIFI network, the user should send the controller his private WIFI password. This is done in the following manner:

1) Upon first power on or after factory reset, the controller boots up as a WIFI hotspot of an unencrypted network.
2) The user is requested to join this network with his smartphone.
3) Using the provided Android or iOS application, the user choose the WIFI network the controller should join, and sends the required password unencrypted (!) over the controller's WIFI.
4) After rebooting, the controller joins the user's chosen encrypted WIFI network using the password it received.

An adversary that can sniff the WIFI communication during the setup process will acquire the user's secret WIFI password and gain access.

A more elaborated attack can allow an adversary to leak the WIFI password even after the setup process. Let's assume the attacker infected a computer inside the the network with a malware. As there is no user authentication for the controller, the malware can change the controller's WIFI in the same manner that the application does using the available API. This will cause the controller to reboot without the ability to connect to the user's WIFI. As the user will have no means to communicate with the controller, he will be forced to do a factory reset and repeat the setup process. The attacker will sniff the setup process and the password. It is worth mentioning that there is no way for the user to differentiate between this attack and any standard malfunction of the controller.

4.1.2. The controller's API. The controller's API is a set of UDP commands sent to the controller's IP or to the networks' broadcast IP. The commands to the bulbs are sent to port 8899, while the settings of the controller are at port 48899. The commands are not encrypted or signed and any member of the WIFI network can change the state of the bulbs.

The commands are sent in the UDP payload as a short series of bytes with a termination byte of 0x55. For example, the light on command is:

IPHEADER	UDPHEADER	0x45	0x00	0x55

and the light off command is:

IPHEADER	UDPHEADER	0x41	0x00	0x55

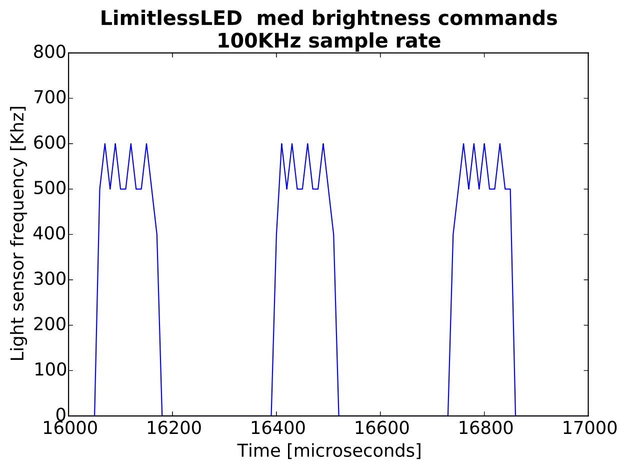


Figure 4. Medium brightness level

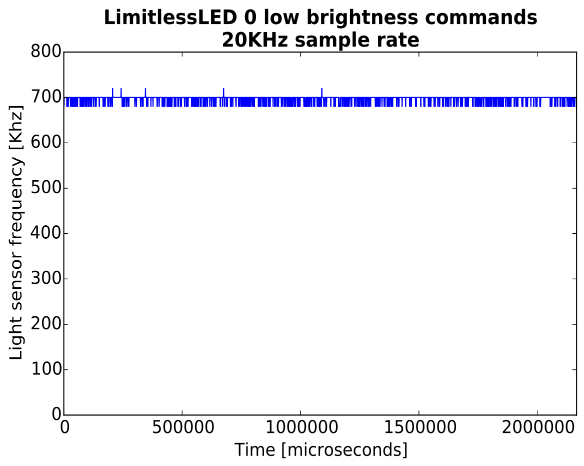


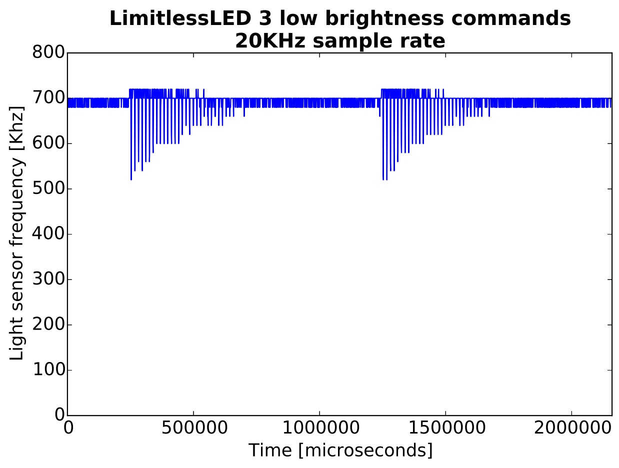
Figure 5. Full brightness

The set brightness command is 0x4e followed by the brightness level between 0x02 (lowest brightness) and 0x1b (full brightness) followed by termination byte 0x55. In particular, the full brightness command is:

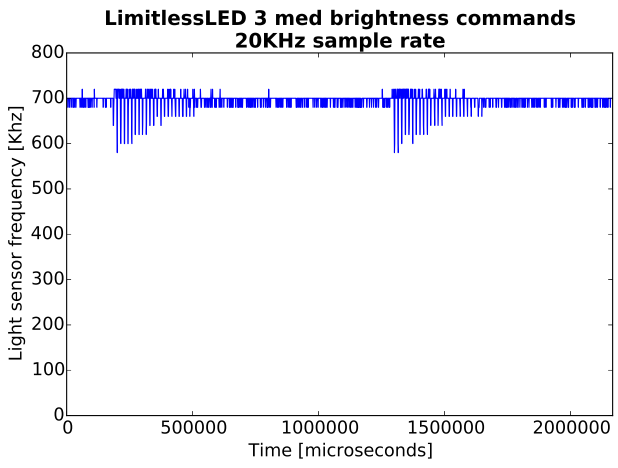
IPHEADER	UDPHEADER	0x4e	0x1b	0x55

Starting from version 3 of the API, the termination byte is optional.

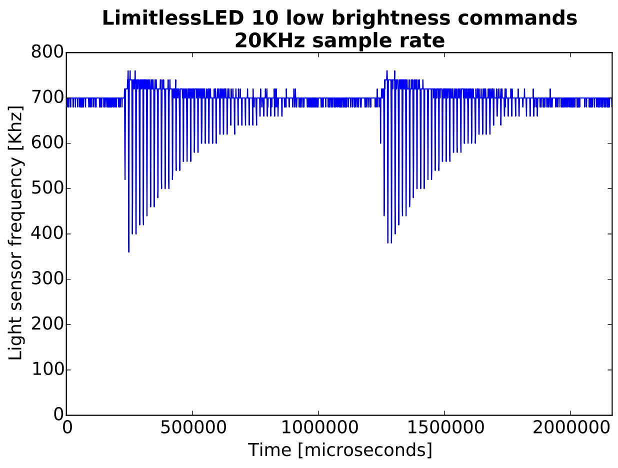
4.1.3. Going over the speed barrier. As in other LEDs, the LimitlessLED bulb controls the brightness by changing the duty cycle of a PWM signal, as seen in Figure 4 for medium brightness. The figure shows the light intensity over time. The light intensity is measured by the frequency outputted by our light sensor (stronger light corresponds to higher frequency). The time is measured in microseconds. Figure 4 is a zoom in on part of the measurement. The high frequency parts correspond to the on period in the duty cycle and the low frequency parts correspond to the off time.



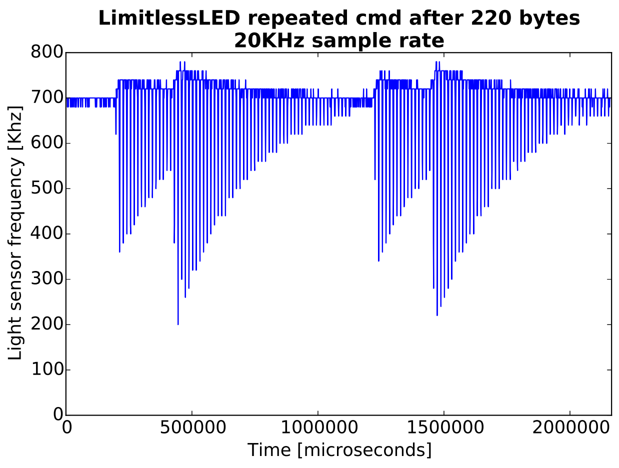
(a) Low brightness - 3 commands



(b) Medium brightness - 3 commands



(c) Low brightness - 10 commands



(d) Low brightness - 10 commands repeated twice

Figure 6. Different commands effects

The LED's drivers and hardware are capable of PWM cycle period of 3 KHz, and very high resolution of duty cycle to support smooth changes between brightness levels. However the controllers API only allow for 27 brightness levels and control rate of less than 10 commands per second or 5 HZ flickers (and is very unreliable at that rate).

After some experimenting we discovered that the controller does not interpret the commands, but only parses the UDP packet and broadcast the payload to the LEDs in the group. The controllers read till the end of the packet or the termination byte 0x55 and send those bytes over the air at a rate of about 1 KBytes/s. When we omitted the termination byte, we were able to concatenate multiple commands up to the MTU (maximum transmission unit - the largest allowed packet size) of the network.

We have seen that decreasing the brightness even by one level causes a visible effect. However, the LED implements a smoothing algorithm when changing brightness levels, slowly changing the PWM duty cycle between the starting and target level. We start from full brightness level (100% duty cycle) as seen in Figure 5. We then send a single command concatenating a long series of low brightness commands followed by high brightness command. That causes the LED to start a smooth decrease in the duty cycle, but to stop and increase it back before the effect is visible to the naked eye.

We can control the signal created by changing the target brightness level (lower target levels create faster decrease in duty cycle). We can also control the exact timing of the change between decreasing and increasing of the duty cycle by the number of low brightness commands we send (repeated commands do not affect the LED, but postpone the timing of the next high brightness command by 2 ms).

In Figure 6a and Figure 6b we can see the different effects caused by a series of 3 low brightness commands followed by high brightness commands relative to 3 medium

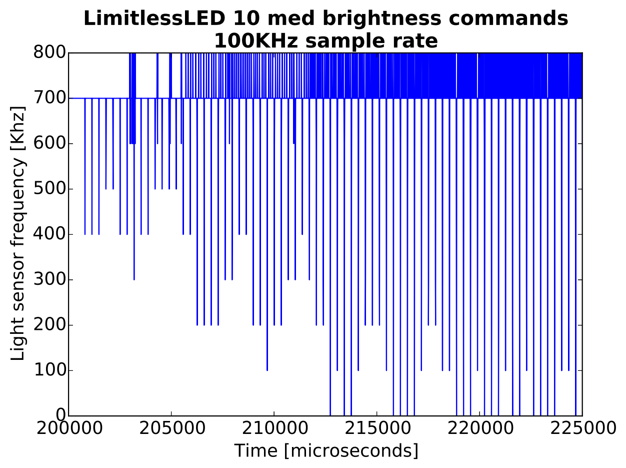


Figure 7. Decreasing brightness effect in high resolution

brightness commands followed by high brightness commands. In Figure 6a and Figure 6c we can see the different effect caused by a series of low brightness commands followed by high brightness commands relative to 10 low brightness commands followed by high brightness commands. In Figure 6d we can see the effect of repeating the command series 2 times in one UDP packet. It enables us to create very accurately timed events. The two drops in intensity are 440 milliseconds apart (220 commands * 2 bytes per command * 1 millisecond per byte).

Using those options we can create different transmitted symbols in the covert channel, which are easily distinguishable by our optical detector, but are not perceivable by the human eye. From those symbols we can reliably read several bits per second (depended on the specific scenario). We can also craft high contrast signals at the relevant frequencies that may induce epilepsy seizures in photosensitive people.

It is worth noticing that our measurements suffer from the same fusion effect of the human eye. Because our sample rate is low, we view off period shorter than our sampling intervals as medium intensity - the average between high and zero intensity. Figure 7 shows this effect. As the duty cycle decreases after a medium brightness command, we pass the point from off time of less than 10 micro seconds (our sampling interval) to more than 10 micro seconds. We see this point as the first time we have intervals of zero intensity. This is not the precise time due to phase differences between our measurements period and the duty cycle period.

4.2. Philips Lux

4.2.1. Going over the speed barrier. Lux provides a much finer control over brightness levels (256 levels), but due to the high frequency rate of the PWM, it is harder to measure those differences. Instead of of relying on the PWM signal generated by the different brightness levels, we used an undocumented API command to generate a PWM signal tailored to our specific covert channel requirements.

In 2012 Philips released a YouTube video [12] showing the effects of an undocumented strobe effect in Hue lights. In 2013 Leon Meijer [13] managed to reproduce the effect using an undocumented option called Points Symbol. This was done by magnifying a computer screen briefly seen in the demonstration video and reading the command. Several different strobing options were found by Meijer and others, but there is little understanding of the way this API works. It was discovered that the first byte of the command controls the speed of the strobe, and some combinations of the other bytes resulted in different colored strobe effects.

We have seen that this option has similar effects in the Lux light, and tried to use it to create fast strobes with different speeds (using the first byte) to create a strobe with small off periods, that will not be visible to the human eye. Unfortunately we discovered that after receiving such a command, the light will blink shortly before starting to strobe, and that causes a visual effect. After some additional trials we discovered that we can send a command in this format (SB is a byte value of the speed and OB is the byte controlling the off period):

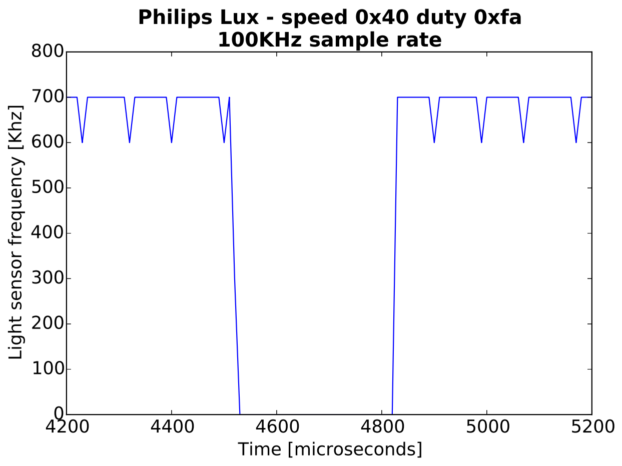
SB	0x00	0xFF	0xFF	0xFF	OB	0xFF
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

As long as we repeatedly sent commands with the same value in the speed byte, there will be no flickering (apart from the first command the can be send on power on to avoid any visible flickering effect). With different off bytes we can create different short off times (lower values mean longer off periods), easily measured by our set up. For speed byte value of 0x40 and off byte of value 0xfa, the off time is 310 microseconds (Figure 8a) and for value of 0xfc the off time is 190 microseconds (Figure 8b).

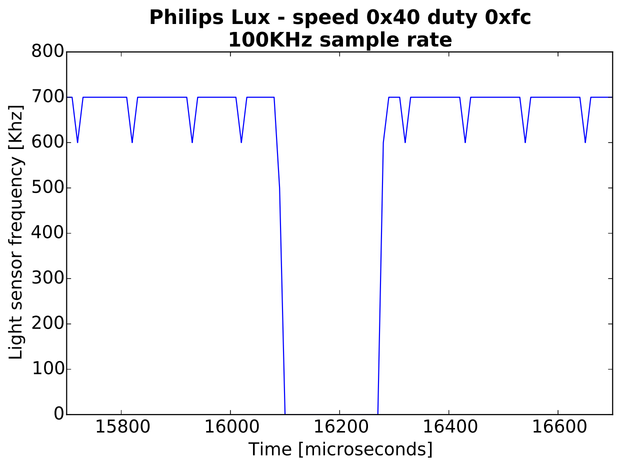
Meijer and others have warned about the risk of epilepsy due to this undocumented strobe effect. The combined effect of poor authentication, our ability to accurately control the strobe timing and duty cycle, and the growing spread of Philips' connected products in public areas such as children hospitals [3] is a big concern. Interestingly Philips had recently announced in their products developers' web page that they will discontinue the support of point symbols from December 2015 due to maintenance difficulties [8].

4.2.2. Leaking data using the Philips Lux's standard API. After this recent announcement by Philips, we were looking for leak methods using only the standard API. We tried again to measure the PWM signal generated by brightness levels changes directly. Philips Lux has 255 different brightness levels, and the difference between two close levels is imperceptible to the human eye. However as stated before, this involves measuring small off intervals of 200 nanoseconds.

To test this we used our light sensor, and a 10 Msa/s (Mega samples per second) analog scope. At the maximum brightness and close range, our light sensor output frequency is around 840Khz which we sampled at 10 MHz. When the brightness level is changed from full brightness (100% duty



(a) 310 microseconds off period



(b) 190 microseconds off period

Figure 8. Different Philips speed effects

cycle) to any lower brightness level, an off period is added, which creates a measurable phase shift in the square wave signal produced by the light sensor. In particular, we were able to differentiate between brightness levels of 255 and 253 at close range. Using more sensitive light sensors and higher rate sampling, we believe that it will be possible to use this method to leak data to similar ranges as our other methods.

5. Results

We created covert channels with the LimitlessLED and the Philips Lux connected LEDs. Using our portable experimental set up, we were able to accurately detect two different symbols from the Philips Lux light from a range of over 100 meters. This building block can be used to covertly leak several bits per second, while using our optical receiver at a safe distance from our target's apartment or office.

The attack we have shown can leak data even from airgapped and Tempest protected networks, with no wireless connections (some connected products can be controlled by wired connections). As lighting in offices is turned on most hours of the day, our slow channel can be used to leak more then 10KB per day. That is enough bandwidth to leak private encryption keys and passwords.

Our attack was implemented using only readily available equipment which can be bought for less then \$1000 on ebay. We used only the available APIs of the controllers, and did not run any unauthorized code on the controllers or LEDs. In addition we did not attempt to preform any reverse engineering or extensive fuzzing.

Using the available APIs we were also able to use both types of LEDs to create strobes of light at frequency ranges that are known to induce seizures in people suffering from photosensitive epilepsy.

6. Conclusions

The results we have shown were due to specific overlooked or unpublished features of the systems. However there will always be new ways of exploiting IoT products to compromise the network, or to use the network to create unintended effects in the physical world. As large organizations will try to use these new solutions they are likely to introduce new security risks, and motivate hackers to find more vulnerabilities.

We believe that it is necessary to focus on security issues during the design, implementation and integration of IoT. For example:

1) IoT designers can use standard security protocols. TLS protocol, with signed certificates that include the specific device name and serial number can be used for the control API. The application can allow the user to verify the specific product he bought. We can use passwords randomly generated by the application and TLS protected channel to maintain the same setup procedure Philips currently employs, while preventing unauthorized entities on the network from eavesdropping or sending commands to the controller. A similar procedure can protect the LimitlessLED setup process.
2) We should limit the IoT API to the necessary minimum. Dropping the undocumented Point Symbol API in Philips products is a step in the right direction. Another example will be to decrease the number of supported brightness levels so that the difference between two close levels will be perceivable by the human observer.
3) The implementation should be pen tested. A simple security code review would have shown that the LimitlessLED controller does not verify the length of the commands he receive.

4) We should consider the way that IoT devices are integrated. We can use Philips' smart light solution for cities [5] as a case study. It is advisable to separate the lights control network from the Internet to protect against attacks such as the blackout attacks suggested in [6]. Connecting the lights to a critical infrastructure network, might expose the networks to attacks such as the ones described here.

Such security measures will make it harder for attackers to exploit IoT devices in the way we described in this paper, but the risk still remains. Researchers and the IoT industry must find ways to improve the security of the IoT systems and networks while maintaining their usability.

References

[1] (2015) Unlocking the potential of the internet of things. [Online]. Available: http://www.mckinsey.com/insights/business_technology/the_internet_of_things_the_value_of_digitizing_the_physical_world

[2] S. Porges. (2012) Forbes magazine
. [Online]. Available: http://www.forbes.com/sites/sethporges/2012/12/28/the-best-product-of-2012-the-philips-hue-led-lighting-system/

[3] Philips. [Online]. Available: http://www.lighting.philips.com/in_en/projects/phoenixchildrenshospital.wpd

[4] S. Higginbotham. (2015) You can introduce the industrial internet with a single light bulb. [Online]. Available: http://fortune.com/2015/04/23/industrial-internet-light-bulb/

[5] Philips. [Online]. Available: http://www.lighting.philips.com/main/cases/cases/parks-and-plazas/hoogeveen-city-center.html

[6] N. Dhanjani. (2013) Hacking lightbulbs: Security evaluation of the philips hue personal wireless lighting system. [Online]. Available: http://www.dhanjani.com/docs/HackingLighbulbsHueDhanjani2013.pdf

[7] Limitlessled technical developer opensource api. [Online]. Available: http://www.limitlessled.com/dev/

[8] Hue's developers program. [Online]. Available: http://www.developers.meethue.com/

[9] [Online]. Available: http://www.zigbee.org/zigbee-for-developers/applicationstandards/zigbee-light-link/

[10] S. Hecht and S. Shlaer, "Intermittent stimulation by light v. the relation between intensity and critical frequency for different parts of the spectrum," *The Journal of general physiology*, vol. 19, no. 6, pp. 965–977, 1936.

[11] W. A. Hershberger and J. S. Jordan, "The phantom array: a perisaccadic illusion of visual direction," *The Psychological Record*, vol. 48, no. 1, p. 2, 2012.

[12] (2012) Philips hue strob effect video. [Online]. Available: https://www.youtube.com/watch?v=CVh4V5QyVQY

[13] L. Meijer. (2013) Do 'hue' want a strobe up there?