universität
innsbruck

Department of Computer Science
Security and Privacy Lab
Course: 703647-0 18S PS3 Netzwerksicherheit

SEMINARTHESIS

Security of Smart Light Systems: Covert Channel on IoT Light Bulb

Bennett Piater 01418470
Julia Wanker 01314695

Innsbruck, June 2018

Preamble

Abstract

Contents

1 Introduction

introduction

2 Related Work

It is widely known that IoT devices have poor security in general. The most recent state-of-the art security survey was performed by Zhang et al. [19] They provide a detailed analysis of vulnerabilities and defense mechanisms. In particular, they note that much academic literature is overly conservative because most security analyses are published in whitepapers and on blogs, causing them to be ignored in scientific surveys. They suspect either a lack of expertise, or outright neglect of security design on the part of the vendors. Additionally, Restuccia et al. [15] recently provided a very good analysis and taxonomy of the systematic problems and future challenges of IoT security. The paper strongly advocates for security by design of connected devices from their cradle to their grave.

The security of smart light systems is particularly important because of their ubiquity. Hence, researchers have studied them in detail. Dhanjani [6] found several ways to initiate Denial-of-Service (DoS) attacks. He was able to cause sustained blackouts which can be of high risk i.e. if hospitals are involved. The primary security issue allowing this attack lay in the connection of smart bulbs to their controller. Dhanjani also mentioned the possibility of encryption flaws in the implementation of the ZigBee Light Link (ZLL) protocol which is used for communication between the bridge and the light bulbs. However, this attack would only work within close proximity, limiting its impact. Morgner et al. [13] further investigated the security of ZLL and showed that the aforementioned attack is more dangerous than anticipated. They were able to control ZLL-certified light bulbs from a distance over 15 to 36 meters. Their research proved and particularized Dhanjani's [6] findings that exploitable vulnerabilities exist in the design of the ZLL standard. ZLL provides the so-called *touchlink commissioning* which uses a global ZLL master key to secure the setup process. This master key was leaked in 2015 [13]

and ever since the touchlink procedure is considered to be insecure. Due to the flaws in the touchlink specification Morgner et al. were able to introduce a new network key which was then accepted by all connected light bulbs, further allowing the authors to send malicious commands. Ronen et al. [17] also used flaws in ZLL to attack smart light solutions. Their attack was of even higher concern since they were able to exchange the light bulbs firmware with one containing malware, and, because of vulnerabilities in the ZigBee communication, they were able to further spread the malware over all nearby light bulbs. Thus, an attacker would be able to launch a *war-flight* and infect all smart lamps of a whole city.

A big portion of the research on IoT security was conducted about attacks ignoring the intended functionality of IoT devices. In particular, the appearance of the Mirai botnet [3] led to multiple papers about botnets comprised of IoT devices. Angrishi [2] makes the very important point that IoT devices should not be seen as specialized devices with added intelligence, but rather as (general) computing devices that are performing specialized tasks. Attackers are certainly aware of this, and most attacks on IoT devices involve botnets for Distributed DoS (DDoS) or spamming. DDoS-capable malware was surveyed and classified by Donno et al. [7]. The most comprehensive analysis of the Mirai botnet was published by Antonakakis et al. [3]. They clearly show that Mirai succeeded primarily because of incredibly low-hanging fruit: (tiny) dictionary attacks on devices accessible from the open internet were enough.

Ronen and Shamir [16] introduced the so called *Functionality-Extending Attack* where e.g. an attacker uses an IoT lightbulb for other purposes than illumination. In particular, an attacker can use light emitting diodes (LEDs) for an optical wireless communication system, which was elaborated several years ago [12, 8]. The communication with visible light is done by repeatedly changing the LED's brightness level. To control the light output from the LED, the pulse width can be modulated, which is known as pulse width modulation (PWM). Using the possibility of creating a communication channel with the help of LEDs, Guri et al. [10] were able to leak data using a router's status LEDs. Since smart light solutions also make use

of LEDs, Ronen and Shamir [16] were able to create a covert communication channel using smart lights. As the setup process of an IoT light bulb is vulnerable [6, 13, 17], Ronen and Shamir were able to abuse the application programming interface (API) of the IoT light bulb in order to make the LEDs switch between two light intensities. To the advantage of an attacker, the brightness of the LEDs can be changed at a very high rate, such that it cannot be noticed by the human eye but can be detected by a light sensor. The light sensor measures the exact duration and frequency of those flickers and converts it to a digital frequency in order to leak sensitive data. Ronen and Shamir showed that this kind of attack can be used to extract data from air-gapped networks and the channel provides enough bandwidth to leak keys and passwords, e. g. in offices where the lights are turned on over 12 hours a day an attacker could leak more then 10 KB per day. Besides leaking data through a covert channel, they have shown that the light flickering can also be misused for creating epileptic seizures.

3 IoT Security

Several former researchers have shown, that security is an afterthought in IoT devices [13, 19, 15, 2, 9, 11]. The main reason for the poor security are the new challenges which arise in the context of IoT systems. Within an IoT system, several different devices can be connected, which affects network performance considerably. Hence also computation and memory capabilities for security measures are limited. Standard public key authentication (PKA) methods, like Rivest–Shamir–Adleman (RSA), are way too expensive due to the provided key size and also approaches with smaller key sizes like elliptic curve-cryptography (ECC) need to be adopted [15]. This cries out for novel security mechanisms specific for IoT devices, but those are not yet realized. Instead, many vendors of IoT devices do not provide proper security measures or even omit them deliberately. On the other hand, access control of remotely sent messages as well as end-to-end message encryption are very important since humans, and thus tons of private information, are tightly involved in IoT systems. If an attacker is able to join ones IoT system, he

could either brick the aimed functionality of the device or use it for different purposes as intended. Attacks on IoT devices can thus be classified as ignoring functionality attacks or extending functionality attacks, which are briefly described in the following sections.

3.1 Functionality Ignoring Attack

In this type of attack the compromised IoT device only serves as a standard computing device within the network since an attacker disregards the usual functionality of the device and instead of that spreads malware over the network the IoT device is in [16]. This attack is indeed not unique to IoT devices, but due to their vulnerabilities, focusing on IoT devices to leverage such an attack might be more promising. In order to ignore the intended functionality of any IoT device, an attacker could initiate a Distributed Denial of Service (DDoS) attack. Further the attacker could spread malware by combining all compromised IoT devices to a botnet. The most prominent DDoS IoT botnet is *Mirai* [7, 11, 3, 2]. In 2016 the Mirai was used to perform a 1.2 Terabit per second (Tbps) great DDoS attack. The attack was relying upon the weak authentication methods of IoT devices.

3.2 Functionality Extending Attack

With this kind of attack, an attacker uses the IoT device for a completely different purpose than intended. This attack is especially conductible on smart light systems, since introducing some unexpected physical effect on other IoT devices is much more sophisticated, i. e. using an IoT fridge or freezer for murder [4]. A functionality extension attack on IoT light bulbs is essentially a covert channel where the communication happens over the light spectrum. The lights frequency output can then be measured by an attacker to exfiltrate sensitive data, which even works from air-gapped networks [16].

4 Communication with Light

Before we can look at how a functionality extension attack on a smart light system can actually be leveraged, we need to understand how communication over light works.

The idea of using visible light as a part of the communication spectrum is known as visible light communication (VLC) [12, 18]. As the name says, the visible part of the electromagnetic spectrum, namely the visible light, is used for communication purposes. VLC is a subset of optical wireless communications technologies, like infrared. In order to conduct VLC, simple white LED bulbs are needed. The actual transmission of data is based on the fact that LEDs can be switched on and off at such a high rate, that those intensity changes cannot be seen by the human eye.

As known from the video game section, human eyes are capable of seeing flickers above 30 Hz. The difference between 30 fps and 60 fps can easily be noticed. Flicker rates which go beyond 60 Hz can though not be detected by the human eye. So in the context of light communication, especially covert light communication as needed in our case, it is important to flicker at over 60 Hz. Those quick flickers allow constant illumination besides the ability of data transmission.

The flickers are done by rapidly switching between on and off states and adjusting the duty cycle. The transmission of data is further allowed by interpreting the on period as logical one and the off period as logical zero. In order to actually get the encoded data out of the light signal the duration and frequency of the light flickers need to be measured. This is done using a light sensor since these are capable of accurately distinguishing between the on and off states and measuring the duty cycle. Further light sensors are robust to other light sources or any other noise.

In the case of smart lights, the PWM signal cannot be changed directly, since the light intensity is changed by sending commands over the manufacturers API. Those commands internally modulate the pulse width of the light signal. Thus, by sending two close brightness commands, we can achieve the same effect as with traditional VLC and covertly transmit data. Different to

traditional VLC, a logical one is represented by the higher brightness level whilst a logical zero is represented by the lower brightness level. Fortunately, since internally the luminosity of the LEDs is again changed by adjusting the PWM signal, light sensors are again capable of measuring the differences.

5 Covert Channel on IoT Light Bulb

Our goal was to reproduce the attack from Ronen & Shamir [16]. Therefore, we tried building a covert channel using the *Philips Hue White* lighting system. With our experiment we were able to prove that data can be covertly transmitted using the Hue light bulb. On the transmission side we used a laptop running our python script over which the Hue API is accessed in order to send brightness commands. At the receiver side we used a light sensor which converts the light intensities to a frequency signal. That signal was forwarded to an oscilloscope which further sent the received frequency output to our laptop where we plot the results in order to validate the sent bits.

In the following sections we first describe the setup components and their functionality. After that, we elaborate the actual attack.

5.1 Experimental Setup

Our proof of concept was implemented using affordable equipment which costs less than 1100.- . Further, we did not need to run any unauthorized code on the light bulbs since the control over the Hue API suffices to create covert flicker effects. Figure 1 shows our overall receiving setup. An Arduino was used as power supply and for configuring the light sensor, which the Picoscope had direct access to.

5.1.1 Transmitting Setup.

We used the *Philips Hue White* light bulbs for our experiment [14]. We bought the starter kit which contains two E27 9 Watt 806 Lumen bulbs together with a bridge which allows to remotely control the bulbs using i. e.

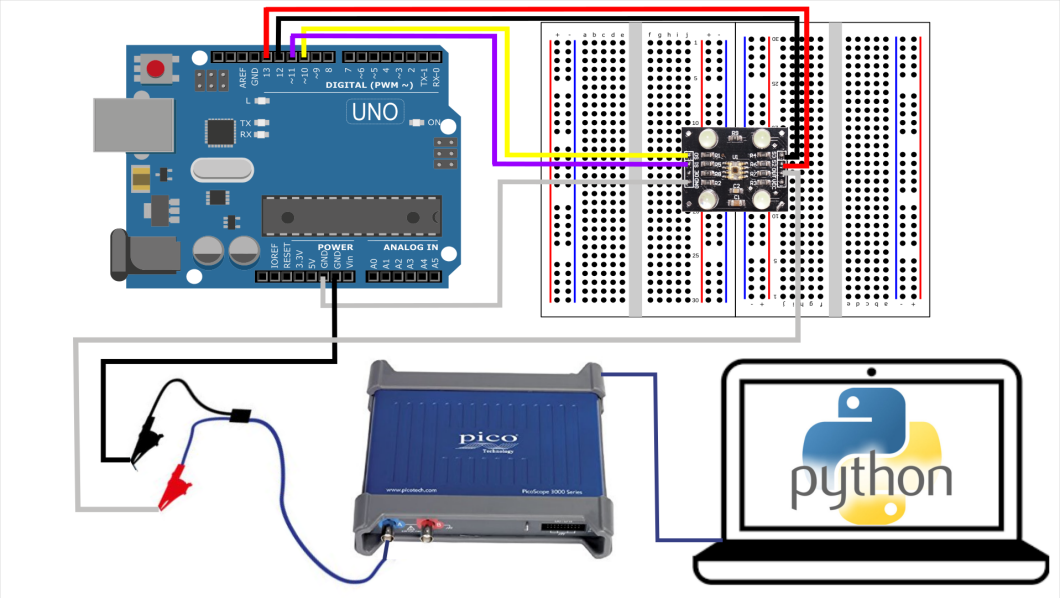

Figure 1: Experimental setup for measuring the lights frequency output and reading the sent bits out of the received signal

a laptop. In order to set up the smart light system the bridge needs to be connected to the user's network using Ethernet. Once connected, the user can send brightness-change commands from any device within the local network using the Hue API. We made recourse to a command line based version of the Hue API [1], which allowed us to easily send the commands via the python script we used for realizing the experiment. The bridge further forwards the commands over a radio frequency (RF) transmitter to the light bulbs using the ZLL protocol.

The Hue White bulbs have 255 different brightness levels, which forced us to sample the output at a very hight rate in order to determine the changes in the light frequency output. But, on the other hand, due to the minor difference between two close levels, the changes were imperceptible to the human eye, which worked for us.

5.1.2 Receiving Setup.

For measuring the changes in light intensity we used the *TAOS TCS3200 Color Sensor* [5]. The sensor consists of an array of photo diodes where each

is capable of filtering red, green, blue or clear white light. We set up the sensor to measure the clear white luminosity output of our Hue bulbs. The sensor contains an internal oscillator in order to convert the light's intensity output to a corresponding square-wave frequency signal. The TCS3200 is capable of communicating directly to an Arduino microcontroller.

Unlike Ronen and Shamir [16], we used the Arduino board as power source only, and the Picoscope to measure the brightness. This was necessary because the advanced API functionality which the former used to craft a PWM signal is no longer available [16], and we had to directly distinguish existing brightness levels using their PWM profile.

In order capture the output from the light sensor, we used the *PicoScope 3205D MSO* since it is capable of sampling 10 MS/s, which we need in order to accurately measure the light sensor's frequency output which lies around 800 KHz. Actually, the picoscope is able to sample up to 1 GS/s when using one channel and 500 MS/s with two channels, but for our needs only 10 MS/s suffice.

5.2 Attack Description

The following paragraphs give a detailed description of the main steps to realize such an attack. Therefore we first need to look more precisely at the functional principle of smart light bulbs. Further we have a look at how smooth brightness changes are achieved and how we actually get data out of the received signal.

5.2.1 Controlling Smart Light Bulbs

Smart light bulbs consist of three main components: (1) a RF receiver, (2) a processing unit and (3) LEDs and LED drivers. The communication with the controller is ensured through the *RF receiver* and relies on the ZLL protocol. The received commands are further forwarded to the *processing unit* which interprets the processed signal and controls the LED by modulating the pulse width. The PWM allows the different dimming factors. When sending a

brightness change command via the Hue API, this automatically forces the processing unit to generate the corresponding PWM signal. The PWM is sent to the LED drivers which further turn the LEDs on and off at a very fast rate such that those changes in the duty cycle cannot be seen by the human eye. Since Hue comes with 255 brightness levels which need to be differentiated smoothly, a PWM with a frequency around 20 KHz is used [16].

5.2.2 Crafting PWM Signal

Since we can no longer craft a custom PWM signal using the Hue API, we had to make do with the PWM used for dimming. Thus, our goal was to use close brightness levels and attempt to distinguish them from their PWM profile.

Because of the great amount of brightness levels, we had to measure very small off periods of about xxx ns. This could be done with the described light sensor as well as the picoscope, since the light sensor's output is around 800 KHz, which we could easily sample at 10 MS/s with our picoscope.

5.2.3 Getting Data

Our primary goal was to distinguish adjacent brightness levels that cannot be distinguished by the human eye. For this purpose, we performed a Short-Time Fourier Transform (STFT) to transform the voltage sinusoid obtained from the light sensor into the frequency-over-time domain.

As can be seen in Figure 2, the difference between brightness levels is clearly visible upon optical inspection of the graph. For comparison, we found that it took around 10–15 levels of separation for the naked eye to distinguish 2 brightness levels, depending on the absolute brightness. In the best images, the PWM profiles were clearly recognizable.

However, we had a rather hard time consistently reproducing images. Distance from the light to the sensor, angle, and variations in external light, in decreasing order, made it harder to clearly distinguish the brightness levels and to choose a starting brightness. In particular, the square curve tended

to become a muddy sinusoid, which we attribute partly to the fact that we perform STFT on the data. We found that distance between sensor and light had a strong impact on the correct choice of STFT window size and intensity of the brightness levels between which we switch, with increasing distance requiring more energy and/or a smaller window.

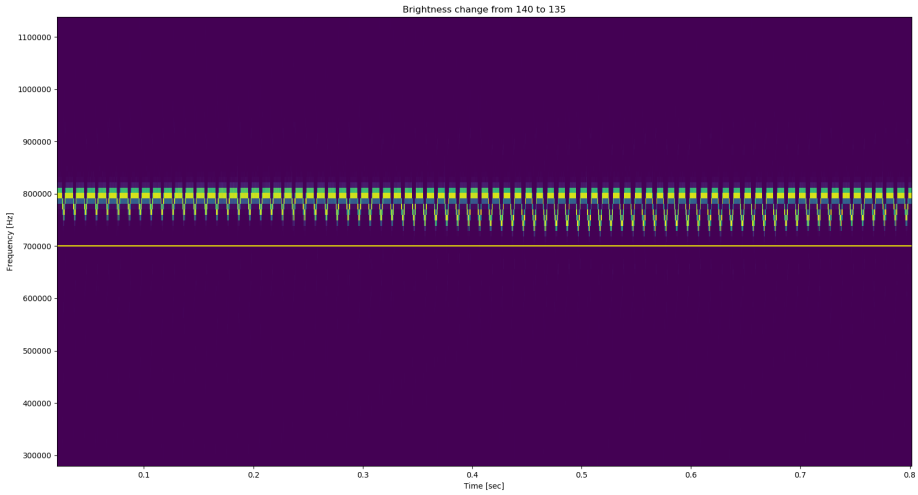

Figure 2: Brightness change from levels 140 to 135 (and back) sampled at 10 MS/s. Note the smooth fading.

We were not able to robustly automatically distinguish brightness levels, but that may be because we spent most of our time trying to do that with the fourier-transformed data when it may have been better to analyze the sinusoid from the light sensor directly. In any case, this is a solvable signal processing problem and more a question of time.

With that done, the next step would be to select brightness levels for 0 and 1, and maybe a third in between the two to be used as delimiter and reduce the need for synchronization. All that would then be left to do is to apply a suitable channel coding to the data, send appropriate brightness commands to the light, write out the recognized 0s and 1s, and decode the data.

5.3 Limitations

Communicating with the light bulbs over the Hue bridge brings some limitations with it [16]. For one, the bridge or the LED drivers implement some smoothing feature in order to avoid sharp brightness changes. Due to the automatic fading we cannot see phase shifts in our signal output, which makes it harder to analyze. Further, the bridge restricts the rate of commands which can be sent within the system. Fortunately, we didn't need to send that much commands for our proof, but in case such an attack should actually be leveraged, one may need to access the ZLL communication directly in order to circumvent the rate limit.

6 Results

describe results of attack

7 Conclusion

work out requirements and summarize results; countermeasures; outlook

References

[1] hueadm - Phillips Hue Admin CLI Utility. GitHub. License: MIT. Online, `https://github.com/bahamas10/hueadm`; accessed 19. June 2018.

[2] K. Angrishi. Turning Internet of Things (IoT) into Internet of Vulnerabilities (IoV): IoT Botnets. *arXiv preprint arXiv:1702.03681*, 2017.

[3] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. Understanding the Mirai Botnet. In Engin Kirda and Thomas Ristenpart, editors, *USENIX Security Symposium*, pages 1093–1110. USENIX Association, 2017.

[4] S. Bhartiya. Your Smart Fridge May Kill You, March 2017.

[5] DFRobot. TCS3200 RGB Color Sensor For Arduino. Website. Online, `https://www.dfrobot.com/product-540.html`; accessed 23. June 2018.

[6] N. Dhanjani. Hacking Lightbulbs: Security Evaluation of the Philips Hue Personal Wireless Lighting System. 2013. Online, `http://www.dhanjani.com/docs/Hacking%20Lighbulbs%20Hue%20Dhanjani%202013.pdf`; accessed 19. June 2018.

[7] M. Donno, N. Dragoni, A. Giaretta, and A. Spognardi. Analysis of DDoS-Capable IoT Malwares. In Maria Ganzha, Leszek A. Maciaszek, and Marcin Paprzycki, editors, *FedCSIS*, pages 807–816, 2017.

[8] H. Elgala, R. Mesleh, H. Haas, and B. Pricope. OFDM Visible Light Wireless Communication Based on White LEDs. In *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*, pages 2185–2189, April 2007.

[9] A. Grau. Can you trust your fridge? *IEEE Spectrum*, 52(3):50–56, 2015.

[10] M. Guri, B. Zadov, A. Daidakulov, and Y. Elovici. xLED: Covert Data Exfiltration from Air-Gapped Networks via Router LEDs. *CoRR*, abs/1706.01140v1, June 2017.

[11] C. Kolias, G. Kambourakis, A. Stavrou, and J. M. Voas. DDoS in the IoT: Mirai and Other Botnets. *IEEE Computer*, 50(7):80–84, 2017.

[12] T. Komine and M. Nakagawa. Fundamental Analysis for Visible-Light Communication System using LED Lights. *IEEE Transactions on Consumer Electronics*, 50(1):100–107, Feb 2004.

[13] P. Morgner, S. Mattejat, and Z. Benenson. All Your Bulbs Are Belong to Us: Investigating the Current State of Security in Connected Lighting Systems. *CoRR*, abs/1608.03732, 2016.

[14] Philips. Hue White Starter Kit E27. Website. Online, `https://www.philips.at/c-p/8718696449554/hue-white-white-starter-kit-e27`; accessed 19. June 2018.

[15] F. Restuccia, S. D'Oro, and T. Melodia. Securing the Internet of Things: New Perspectives and Research Challenges., March 2018.

[16] E. Ronen and A. Shamir. Extended Functionality Attacks on IoT Devices: The Case of Smart Lights. In *EuroS&P*, pages 3–12. IEEE, 2016.

[17] E. Ronen, A. Shamir, A. Weingarten, and C. O'Flynn. IoT Goes Nuclear: Creating a Zigbee Chain Reaction. *IEEE Security & Privacy*, 16(1):54–62, 2018.

[18] Z. Yu, R. J. Baxley, and G. T. Zhou. Brightness Control in Dynamic Range Constrained Visible Light OFDM Systems., January 2014.

[19] N. Zhang, S. Demetriou, X. Mi, W. Diao, K. Yuan, P. Zong, F. Qian, X. Wang, K. Chen, Y. Tian, C. A. Gunter, K. Zhang, P. Tague, and Y. Lin. Understanding IoT Security Through the Data Crystal Ball: Where We Are Now and Where We Are Going to Be., March 2017.