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

S E M I N A R T H E S I S

IoT Light Bulb Covert Channel

Bennett Piater	**0xxxxxxxx**
Julia Wanker	**013146957**

Innsbruck, June 2018

Contents

1 Introduction

introduction

2 Related Work

related work chapter

3 IoT Security

IoT security introduction to chapter, short description of state of the art

3.1 Ignoring Functionality Attack

functionality ignoring attacks description

3.2 Extending Functionality Attack

functionality extending attacks description

3.3 IoT Light Bulbs

functionality extending attacks described for light bulbs in more detail

4 Related Topics

description of required knowledge in order to be able to understand and conduct attack

4.1 Covert Channel

describe how covert channel work

4.2 Communication with Light

describe general communication with light

5 Covert Channel on IoT Light Bulb

We have created a covert communication channel on the *Philips Hue* light bulb by changing near brightness levels at a very high rate such that those changes cannot be seen by the human eye but can be robustly distinguished by a light sensor. To prove that data can be leaked over this channel we used a oscilloscope. In the following sections we first describe the setup components and their functionality. After that we elaborate the actual attack.

5.1 Experimental Setup

5.1.1 IoT Light Bulb.

We used the *Philips Hue White E27* light bulbs [?]. The bulbs come together with a Wi-Fi bridge which allows to remotely control them using i. e. a laptop. In order to setup the smart light system the bridge needs to be connected to the user's local Wi-Fi or 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 [?], which allowed us to easily send the commands via the python script we used for realizing the data transmission. Hue has 255 different brightness levels, which forced us to sample at a very hight rate in order to determine changes in the light frequency output. The difference between two close levels is imperceptible to the human eye but can properly be detected by a light sensor.

5.1.2 Light Sensor.

For measuring the changes in light intensity we used the *TAOS TCS3200 Color Sensor* [?]. This sensor gives the corresponding frequency output to the received light intensity. Further this sensor can easily be used with Arduino.

5.1.3 Arduino.

Other than former researchers [1], we used the Arduino board as power source only, since the frequency output can more easily be measured using the *PicoScope* described below.

5.1.4 Picoscope.

To decode out our covert channel we used the *PicoScope 3205D MSO* since it is capable of sampling 10 MS/s, which we need in order to measure the light sensor's frequency output. Actually, the PicoScope is able to sample up to 1 GS/s, but for our needs 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 Contorlling Smart Light Bulbs.

The communication with the controller is ensured through a *RF receiver* and relies on the *ZigBee Light Link* (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 [1].

5.2.2 Crafting PWM Signal.

5.2.3 Getting Data.

6 Results

describe results of attack

7 Conclusion

work out requirements and summarize results

References

[1] Eyal Ronen and Adi Shamir. Extended functionality attacks on iot devices: The case of smart lights. In *EuroS&P*, pages 3–12. IEEE, 2016.