Frying PAN: Dissecting Customized Protocol for Personal Area Network *

Kibum Choi[1]✉, Yunmok Son[1], Jangjun Lee[2], Suryeon Kim[2], and Yongdae Kim[1,2]

[1] School of Electrical and Engineering, KAIST, Republic of Korea
[2] Graduate School of Information Security, KAIST, Republic of Korea
{kibumchoi,yunmok00,baator.nine,c16192,yongdaek}@kaist.ac.kr

Abstract. A spoofing attack for a wireless communication system is the most common attack method for unauthorized access and control. IEEE 802.15.4 is a standard that defines only physical and medium access control layers for low rate, low power, and low cost wireless systems. This standard is widely used as lower layers for not only several wireless communication standards but also customized protocols by manufacturers. However, security has not been considered seriously in these customized protocols, due to other important features including efficiency and cost. In this paper, in order to empirically analyze the real world threat in these systems, we chose to study three IEEE 802.15.4 based wireless communication systems as targets. We manually analyzed the customized protocols above IEEE 802.15.4 if there exist vulnerabilities to be exploited. For all three systems, we discover significant vulnerabilities. We implemented a spoofing attack for two targets, and we successfully controlled the targets by our spoofing attack. For the last target, we chose not to run the experiment due to significant safety reasons.

Keywords: Security, Wireless spoofing attack, IEEE 802.15.4

1 Introduction

IEEE 802.15.4 is a standard that specifies the physical (PHY) and the medium access control (MAC) layer for Low-Rate Wireless Personal Area Networks (LR-WPAN) [1]. It operates on one of three frequency bands, 868-868.6MHz, 902-928MHz, or 2,400-2,483.5MHz. IEEE 802.15.4 can provide up to 250kbps at 10m distance with low power and low cost. Due to the requirement of efficiency in power and cost, IEEE 802.15.4 is designed as a simple structure, and does not readily to support encryption or authentication.

Despite its absence of security features, IEEE 802.15.4 is still utilized in the field of Internet of Things (IoT) devices, such as wireless sensor networks, smart home network and industrial controlling systems, on the basis of its efficiency.

* This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. NRF-2014M3C4A7030648)

Numerous wireless communication systems such as ZigBee, 6LoWPAN, WirelessHART, and MiWi have been built upon IEEE 802.15.4.

Notably, we found three critical applications that are using custom protocols upon IEEE 802.15.4. Our first target, Smart plug, is a smart power metering system. It can turn on and off a power supply, and collect the power usage information of users. The second target is a door lock system controlled by both its user and a manager. By the user's request or in an emergency situation, the manager can control the door lock system remotely using wireless communication. The last target is a Platform Screen Door (PSD) system that communicates with a control system using a wireless protocol in subway stations. Although, these systems are closely related to security and safety, they are implemented without any security concerns and only rely on their custom protocol.

In this work, we captured packets from three target systems using commercial RF transceivers. By analyzing these packets manually, it was possible to infer most fields of customized protocols, and we then implemented a spoofing attack to take control of them. We were able to successfully control the first and second targets, but we could not perform our an attack on the third, because of legal and safety issues. Note that all vulnerabilities are responsibly disclosed in advance.

To summarize, we made the following contributions.

- We derived general analysis methodology for customized protocols on top of IEEE 802.15.4, which is far different from that of TCP/IP packet analysis.
- We found nine security vulnerabilities from three real world targets using the proposed analysis methodology. Furthermore, we were able to exploit these vulnerabilities to spoof the protocol messages.
- We performed unauthorized control of critical applications which are linked directly with the citizen's safety.

The remainder of this paper is organized as follows. Section 2 describes related work on the attack of IEEE 802.15.4 networks. Section 3 provides background on IEEE 802.15.4 and tools for the spoofing attack we used. Sections 4 and 5 explain our analysis methodology and generalized vulnerabilities of customized protocols over IEEE 802.15.4. Section 6 presents a detailed analysis and attack against two target systems and the result of the attacks are presented in Section 7. Section 8 concludes the paper.

2 Related Work

Several works on attacking IEEE 802.15.4 based systems in both the PHY and MAC layers have been reported. While most PHY layer attacks are related to jamming that causes a denial of service for systems, our main concern is taking control or causing malfunctions against the target system by injecting well crafted packets. Spoofing attacks are usually related to upper layers rather than the PHY layer. Some papers related to security on the MAC layer of IEEE 802.15.4 systems have been published. Sastry et al. mentioned three problems that could reduce security in IEEE 802.15.4 specifications [2]. First, there is an

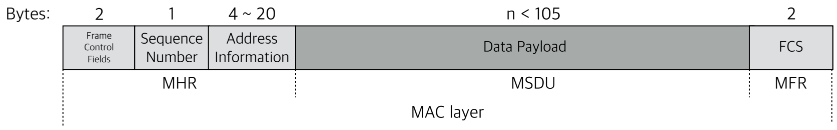


Fig. 1. IEEE 802.15.4 Data Frame Format

initial vector (IV) management problem where the same key can be used in two different access control list (ACL) entries. The second problem is no support for group keying because each ACL entry can only be associated with one destination address. The third problem is that the standard supports an unauthenticated encryption mode that is vulnerable. Sokullu et al. suggested a Guaranteed Time Slots (GTS) attack in the IEEE 802.15.4 MAC layer [3]. GTS is a part of the superframe for collision-free transmission, and thus they are exclusively dedicated to a single device. An attacker can learn the GTS slot times from the beacon frame following a GTS request, and then she can cause interference or a collision when a legitimate node transmits a GTS data frame. A GTS attack is a denial of service against GTS requests. Jokar et al. presented Received Signal Strength (RSS) based spoofing detection and prevention techniques in static IEEE 802.15.4 networks [4, 5].

Most related works are focused on the security of the IEEE 802.15.4 standard itself. In contrast to previous works, we point out that customized protocols on IEEE 802.15.4 are utilized for actual devices and they have security problems.

3 Background

In this section, we explain the basic information of the IEEE 802.15.4 architecture, data frame format, and platforms used for our spoofing attack.

IEEE 802.15.4 Architecture The IEEE 802.15.4 architecture consists of only two layers, the PHY layer and the MAC layer. The PHY layer defines the physical specifications to transmit and receive radio frequency (RF) signals through a physical transmission medium. The MAC layer provides a reliable link between two nodes, and is responsible for encoding digital bits into packet frames to transmit, decoding them to receive frames, and controlling the access to data in a network. This architecture is utilized not only for the lower layer of several wireless communication standards but also for unknown customized protocols designed by device developers.

Data Frame Format Figure 1 depicts a data frame format of the MAC layer. The MAC frame contains the MAC Header (MHR), MAC Service Data Unit (MSDU), and MAC Footer (MFR). The MHR has a Frame Control Field (FCF), data sequence number, and address information. MSDU is the actual data to transmit and MFR is Cyclic Redundancy Check (CRC) for error detection.

In this work, the most important field is MSDU, because it carries a payload that is composed of the customized protocol data for the specific purpose or service of various manufacturers. Furthermore, FCF, the first two bytes in MHR, contains important information that is represented as a bit-map such as packet type, security enabled status, ACK request status, and addressing mode. If a security enabled field is '0', then data is not encrypted. This is the starting point of protocol reverse engineering.

Attack Platforms We used two programmable RF transceiver devices to collect and inject IEEE 802.15.4 packets as attack platforms. The first is Universal Software Radio Peripheral (USRP) with "gr-ieee-802.15.4" GNU Radio module that fully supports the IEEE 802.15.4 standard. Another platform is KillerBee [6] which is a python based framework designed for the analysis of IEEE 802.15.4 and ZigBee protocol. By using these hardware and software platforms, we can obtain data frames of the MAC layer regardless of the PHY layer protocol. In addition, log files from these platforms are readable by Wireshark.

4 Methodology

Our analysis can be divided into three phases to make a spoofing attack possible for an IEEE 802.15.4 based customized protocol: collecting, grouping, and actual analysis. To understand the fields of a customized protocol as well as possible, it is necessary to control the variance of packets in known or predictable conditions in the first two phases. We strongly believe that this methodology with three phases can be considered as a generic approach for IEEE 802.15.4 based customized protocol reverse engineering.

4.1 Collecting Packets

The first step to take for a sniffing attack is to find a communication channel. By brute-forcing channels, we can find the active channel that the target system uses. The collecting phase is usually considered as a simple process, as it is supported in the hardware we use. However, the challenges in the next phases depend on this phase. There are variable factors or environments that feasibly affect the variance of the packet data such as function, date, timing, and place. Note that not all of these are necessary for building packets and all implementations are somewhat different.

Function is the most basic factor to distinguish command related fields, and date or time is related to the timestamp field. Sometimes it is necessary to capture packets within a limited time span to remove unnecessary packets that may not include critical information. Place is also an important factor, because the network topology can be one master to one client, one master to multiple client, or multiple masters to multiple clients. To control and identify the source and destination address fields among multiple communication pairs, the physical location should be addressed.

Both fixing and changing these factors may be necessary to identify the related fields in packets. Therefore, an attacker must control variable factors as much as possible.

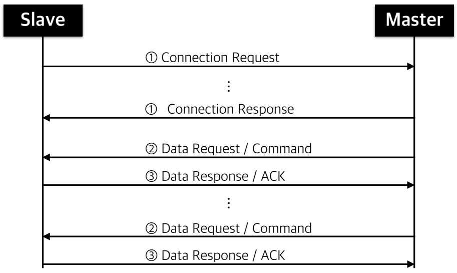


Fig. 2. IEEE 802.15.4 Communication process in three steps

4.2 Grouping Packets

The second phase is a grouping process for the collected packets. There are many types of packets in a common digital communication protocol such as request, response, command, acknowledgement, and so on. Because the formats of these various types usually have different fields, they make the protocol analysis more difficult. By classifying the collected packets according to the source or destination addresses, we can learn the information of request to response and command to acknowledgment relationships, which are shown in figure 2. The length of the packet can provide useful information, because the different fields lead to different packet lengths. For example, acknowledgement packets are usually very short for efficiency. It is also helpful to categorize the collected packets according to the factors mentioned in the previous phase.

4.3 Analysis Protocol

The last step is the actual analysis process of hex or ASCII valued byte data in the packets. In the case of customized protocols, most of this phase is conducted manually. The important elements in the sequence of grouped packets are the repeated data, periodic data, monotonically increasing or decreasing data, and other meaningful data. The repeated data can be related to the control features of the two previous phases including source or destination addresses, functional commands, and so on. Periodic data also can be interpreted to have various meanings by their timing characteristics such as the interval. Sequence number and timestamp of packets usually appear as monotonically increasing data, and they can be decoded according to their cycles. For example, the cycles of date or time data can be 12, 24, or 60 in decimal or hex, whereas that of sequence number can be a full byte size. In particular, repeated data in every packets without any change are important, because an attacker can use them as fixed values for a spoofing attack without detailed knowledge.

5 Vulnerabilities

During the protocol reverse engineering against three targets, we found several vulnerabilities in those customized protocols. We found features that in fact helped our protocol analysis. We listed these problems in table 1.

Table 1. Vulnerabilities in custom protocols based on IEEE 802.15.4

Phase	Vulnerability	Smart Plug	Door Lock	PSD
Analysis	No encryption on payload	✓	✓	✓
	No packet fragmentation	✓	✓	✓
	Plaintext data (ASCII)	✓	✗	✓
	Repeated MAC layer data	✓	✓	✓
	Periodic increase (timestamp)	✗	✓	✓
	Sequential increase (seq num)	✓	✓	✓
Spoofing	Well-known CRC	✓	✓	✓
	Meaningless fixed field	△	✓	✓
	Poor authentication	✓	✗	△

5.1 Vulnerabilities in Analysis Phase

The most fundamental vulnerability in WPAN is the absence of encryption on the payload. This is the starting point of our protocol reverse engineering. When we analyzed the customized protocol, packet parsing was our basic approach. From an attacker's perspective, if packets are fragmented, then parsing is infeasible. If a packet shows plain text in ASCII, then we can easily infer the meaning of the packet or field. Even for packets that use non-ASCII range bytes, repeated byte fields were often detected. All three customized protocols use modified parts of the MAC address in their address fields, probably because using new address requires address to device binding. This would be an easy way to identify the source and destination field.

5.2 Vulnerabilities in Spoofing Phase

For a successful spoofing attack, normally we had to control two values, the sequence number and the CRC. The remaining fields were only copied from previous packets or could be hard-coded. If some fields use fixed bytes for every operation, then we do not have to guess the real meaning of fields. In addition, all three targets used well-known CRC methods. Moreover, some of them only check if the sequence number is larger than the current sequence number.

6 System Analysis

We now present our analysis following methodology consists of collecting packets, grouping packets, and protocol reverse engineering phases. This section contains three different customized systems based on IEEE 802.15.4.

6.1 Smart Plug Analysis

A. Smart Plug System Overview

Our first target is a wireless smart power metering system. This system consists of a Watt Checker Unit (WCU) and a Data Concentrate Unit (DCU). The WCU is set up between home appliances and a wall outlet, and it measures

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
ID	CMD	sequence number				source address								destination address								PAN ID		ASCII representation
53	4F	32	31	31	34	17	04	00	00	00	55	43	44	64	0A	00	00	00	55	43	57	04	17	SO2114.....UCDc.....UCW..
53	61	32	31	31	35	17	04	00	00	00	55	43	44	64	0A	00	00	00	55	43	57	04	17	Sa2115.....UCDc.....UCW..
53	46	32	31	31	36	17	04	00	00	00	55	43	44	64	0A	00	00	00	55	43	57	04	17	SF2116.....UCDc.....UCW..
53	61	32	31	31	37	17	04	00	00	00	55	43	44	64	0A	00	00	00	55	43	57	04	17	Sa2117.....UCDc.....UCW..

Fig. 3. Packet format of smart plug on/off command

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
ID	CMD	sequence number				source address								destination address								stat	ASCII representation
57	73	32	30	39	32	63	0A	00	00	00	55	43	57	17	04	00	00	00	55	43	44	31	Ws2092c....UCW.....UCD1
57	73	32	30	39	33	63	0A	00	00	00	55	43	57	17	04	00	00	00	55	43	44	30	Ws2093c....UCW.....UCD0

Fig. 4. Packet format of smart plug on/off ACK

power usage data in real time and reports the data to the DCU. The DCU collects data from multiple WCU and displays the data. It can also remotely control each WCUs to turn it on or turn off.

B. Smart Plug System Analysis

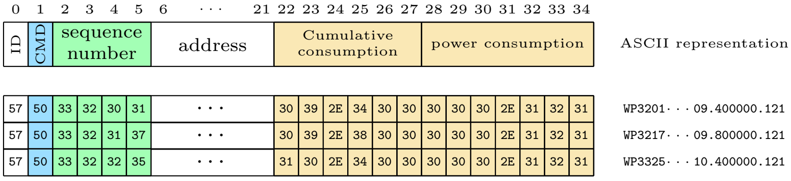
Packet capture First, it was important to identify the communication channel to sniff packets. Because there are only 16 channels in the 2,4GHz bands, we could easily brute force this. Initially, we found the source and destination address, packet length, and data type from the header fields of captured packets. Because the security option is disabled, we started reverse engineering.

On/Off command One on/off operation consists of three packets, the DCU on/off command request, response from WCU, and DCU ACK. After we found communication process, we sorted packets into two sets with address field and packet length. Figures 3 and 4 represent the arranged data from the two sets and protocol reversing results. After the sorting step, we had to cut packets into meaningful units. We used our analysis methodology in section 4

We identified the address fields that are constantly repeated and have exactly the same values of source and destination address of header fields. Byte fields at positions 6-13 and 14-21 have exactly the same values of source and destination address, respectively.

Among the remaining bytes, bytes 1 and 5 are the only changing fields. In particular, byte 5 was changed from `0x30` (‘0’ in ASCII) to `0x39` (‘9’ in ASCII) circularly like a counter. After scanning numerous packets for long period, we found periodicity in 4 bytes from byte 2. From this, we determined that field to be the sequence number field with a decimal number.

The most important byte was byte 1 which prints ‘O’ and ‘F’. Most of the fields used ASCII formats in the captured packets, and we could infer that ‘O’ and ‘F’ represent “ON” and “OFF”. Likewise, ‘a’ may represents “answer” or “acknowledgment”.



0	1	2	3	4	5	6 · · · 21	22	23	24	25	26	27	28	29	30	31	32	33	34	
ID	CMD	sequence number				address	Cumulative consumption						power consumption							ASCII representation
57	50	33	32	30	31	· · ·	30	39	2E	34	30	30	30	30	30	2E	31	32	31	WP3201· · · 09.400000.121
57	50	33	32	31	37	· · ·	30	39	2E	38	30	30	30	30	30	2E	31	32	31	WP3217· · · 09.800000.121
57	50	33	32	32	35	· · ·	31	30	2E	34	30	30	30	30	30	2E	31	32	31	WP3325· · · 10.400000.121

Fig. 5. Packet Formatof Smart Plug Power Consumption Data

Sorted packets in the figure 4 are the WCU response for power control to the DCU. It has a similar format to that of the DCU. As a command in byte 1, the WCU uses 's', which may mean "success". The last byte in the WCU alternatively prints '0' and '1' in ASCII. This last byte likely means the power on/off status of the WCU. Another difference between the WCU and DCU formats is byte 0. The DCU uses 'S' and the WCU uses 'W' instead of 'S'. However, for a spoofing attack, fixed bytes were not our concern.

Power monitoring command The DCU requests power consumption information to WCUs every minute. Like the power control operation, the power monitoring process is also divided into three steps. The DCU requests power consumption information, and then the WCU replies with power consumption information. Finally, the DCU sends an acknowledgment to the WCU. In this process, byte 1 changes as follows: 'P', 'P', and 'a'.

Power monitoring packets have almost an identical structure with the power control packet structure up to byte 21. The DCU only adds PAN ID at the end of the power request and ACK packets. The remaining bytes in the WCU have power consumption information. Two key features that we could understand the meaning of are the range of hexadecimal bytes and repeated bytes such as `0x30` ('0' in ASCII) and `0x2E` ('.' in ASCII). From these, we determined that this field is using ASCII formats to represent cumulative power consumption and instant power consumption.

6.2 Door Lock Analysis

A. Door Lock System Overview

The second target is a wireless controlled digital door lock system. This system consists of a door lock, a transceiver and a central controller. When the central controller sends a door open signal to the transceiver, which is installed in each floor, the transceiver sends a wireless packet to a specific door lock.

B. Door Lock Protocol Analysis

Packet capture In an apartment, the distance between a door to another door is close enough to receive packets in the same floor. After we found a channel, we could collect all packets between a transceiver and multiple door locks. However, since the door opening operation from the central controller is not a normal operation, we operated KillerBee until we collected enough packets.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ID	Seq	CMD	Lower 32bit of src MAC addr				Prefix (00:15:8d:00)				Lower 32bit of dst MAC addr				Prefix (00:15:8d:00)				Misc.				END

Fig. 6. Packet Format of Door Lock Open Command

Door status monitoring command After gathering bunch of packets, we grouped packets according to packet length. We found six different length packets. We create a variety of situations to classify the different size of packets for further analysis. Majority of them are door status monitoring packets that consist of 2 phases. Firstly, request command packet of 47 byte length is transmitted from controller to door lock. Secondly, response command of 48 byte length is transmitted from door lock to controller. Packet of 47 bytes has a same feature with door open command which will be discussed right below. We disintegrated response packet of 48 bytes. Most of field was fixed making our work much easier. According to the change of command type between request and response the value of byte 0 repeats `0x02` and `0x05`. Four bytes from byte 18 represent time field consisting of `mm/dd/hh/mm` meaningful as a status information. Each request and response command has a subsequent ACK packet of 5 byte length. ACK packet is simply configured with FCF, sequence number and CRC. The above procedure repeats every 5 minutes between controller and door locks.

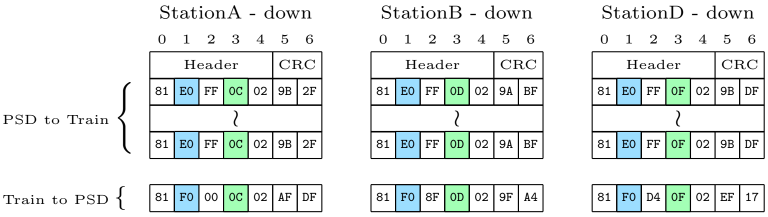
Door open command With repeated door open and close experiments, we could filter out status monitoring packets from the door lock to the central controller. In figure 6, we dissected the format of the door lock open command. From byte 3 to byte 18, we found that address fields consist of modified MAC layer addresses. The last byte is always fixed when packets have different lengths. We assumed that this field means the end of data. The remaining fields are variable fields. The value of byte 0 repeats `0x02` and `0x05` corresponding to request and response. Obviously, byte 1 is a sequence number. When the byte length is changed, byte 2 is changed. Therefore, it shows the type of packets or the kind of command code. Four bytes from byte 19 are important. In normal cases, it receives `0x02013B3B`. When the door is opened with a signal sent from the central controller, then `0x02014848` is captured.

6.3 Platform Screen Door System Analysis

A. PSD System Overview

PSDs have been installed at subway platforms to separate the platform from the subway train for the safety of passengers and a pleasant air environment of subway stations. For citizen's safety, the PSD design must be secure. We found customized IEEE 802.15.4 based wireless communication in PSD installed in the subway station of the largest city of our country. The PSD system coincides with figure 2. The PSD controllers installed in a train and a station correspond to a master and slave, respectively. We found two subway lines use a customized protocol based on IEEE 802.15.4 and successfully reverse-engineered it.

B. PSD System Analysis



StationA - down

	0	1	2	3	4	5	6
	Header					CRC	
PSD to Train	81	E0	FF	0C	02	9B	2F
	≀						
	81	E0	FF	0C	02	9B	2F
Train to PSD	81	F0	00	0C	02	AF	DF

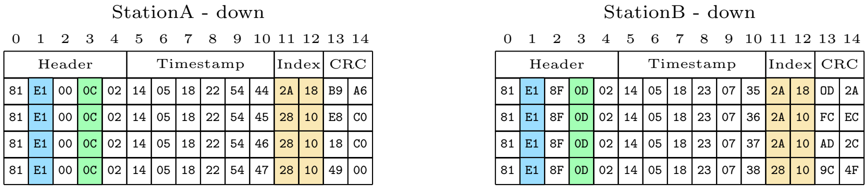
StationB - down

	0	1	2	3	4	5	6
	Header					CRC	
PSD to Train	81	E0	FF	0D	02	9A	BF
	≀						
	81	E0	FF	0D	02	9A	BF
Train to PSD	81	F0	8F	0D	02	9F	A4

StationD - down

	0	1	2	3	4	5	6
	Header					CRC	
PSD to Train	81	E0	FF	0F	02	9B	DF
	≀						
	81	E0	FF	0F	02	9B	DF
Train to PSD	81	F0	D4	0F	02	EF	17

Fig. 7. PSD packet format - step 1



StationA - down

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Header					Timestamp						Index		CRC	
81	E1	00	0C	02	14	05	18	22	54	44	2A	18	B9	A6
81	E1	00	0C	02	14	05	18	22	54	45	28	10	E8	C0
81	E1	00	0C	02	14	05	18	22	54	46	28	10	18	C0
81	E1	00	0C	02	14	05	18	22	54	47	28	10	49	00

StationB - down

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Header					Timestamp						Index		CRC	
81	E1	8F	0D	02	14	05	18	23	07	35	2A	18	0D	2A
81	E1	8F	0D	02	14	05	18	23	07	36	2A	10	FC	EC
81	E1	8F	0D	02	14	05	18	23	07	37	2A	10	AD	2C
81	E1	8F	0D	02	14	05	18	23	07	38	28	10	9C	4F

Fig. 8. PSD packet format - step 2

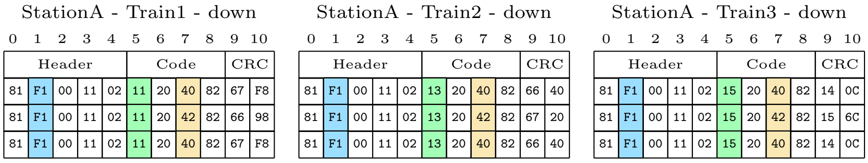
PSD Packet Capture As a train approaches a platform, we try to detect an active channel by brute forcing. Fortunately, the train and the PSD exchange packets with a 20ms interval for almost 2 to 3 seconds. Therefore, we scanned all 16 channels by changing the receiving channel for each 100ms. Furthermore, the communication channel is fixed for the same line and for the same direction. We systematically collected packets against 20 subway stations and several trains to reverse the PSD protocol.

PSD System Protocol Reversing for Line X and Y First, we analyzed captured packets from line X, due to the completeness of the captured packet sequences. (Line Y has an identical packet structure with line X). First, we verified the header of the MAC layer to obtain meta data. The value of FCF is `0x8841`, which represents a disabled security option, no ACK, and a 16-bit source and destination address. We grouped PSD packets in three steps following figure 2.

① Step 1 - Connection Request and Response Packets corresponding to step 1 from three different stations are depicted in figure 7. In the whole process, we found that a header field is 5 bytes.

Byte 1 represents the communication step. We could see the transition of byte 1 when the communication process changes. Byte 1 from PSD, `0xE0` in step 1 changed to `0xE1` in step 2 packets. Otherwise, byte 1 from the train changed from `0xF0` in step 1 to `0xF1` in step 3. Bytes 2 and 3 describe the lower 8 bits of the train MAC address and the station MAC address (each address is 16-bits), respectively. Thus, `0xFF` in byte 2 indicates broadcasting address. We could also observe the increment in byte 3 when the train moved to the next station. Byte 4 is always `0x01` or `0x02`. After numerous experiments in the same station, we found that byte 4 represents the direction of the train. Bytes 5 and 6 contained CRC-16, which is a well-known protocol.

② Step 2 - Data Request of PSD In figure 8, the packet consists of a header, timestamp, and CRC. Byte 1 changed to `0xE1` from `0xE0` in step 1. We



StationA - Train1 - down

0	1	2	3	4	5	6	7	8	9	10
Header					Code				CRC	
81	F1	00	11	02	11	20	40	82	67	F8
81	F1	00	11	02	11	20	42	82	66	98
81	F1	00	11	02	11	20	40	82	67	F8

StationA - Train2 - down

0	1	2	3	4	5	6	7	8	9	10
Header					Code				CRC	
81	F1	00	11	02	13	20	40	82	66	40
81	F1	00	11	02	13	20	42	82	67	20
81	F1	00	11	02	13	20	40	82	66	40

StationA - Train3 - down

0	1	2	3	4	5	6	7	8	9	10
Header					Code				CRC	
81	F1	00	11	02	15	20	40	82	14	0C
81	F1	00	11	02	15	20	42	82	15	6C
81	F1	00	11	02	15	20	40	82	14	0C

Fig. 9. PSD packet format - step 3

found the timestamp field was formed as `yy/mm/dd/hh/mm/ss` from the fact that the period is 60 in decimal. The index field changed following the status of the PSD. We assumed that field to be an index code for the purpose of PSD status monitoring.

③ Step 3 - Data Transmission of Train Figure 9 represents packets from train to the PSD. Byte 2 changed to `0xF1` from `0xF0`. In the code field, bytes 6 and 8 are never changed. Although we could not find the meaning of the bytes, we can fill these fields with the same values for the spoofing attack. However, byte 5 increased by two as the next train entered the platform. Since the PSD closed when byte 7 changed to `0x42` from `0x40`, we speculated that it was the practical PSD control code. This field is the only changing field of step 3. Thus, we believe byte 7 is a control message.

7 Attack Results

7.1 Spoofing Smart Plug

To implement a spoofer, we utilized the KillerBee framework. The only task that we need to perform is a simple payload forgery. We hard-coded constant fields such as addresses and inserted the command. We additionally attached sequence number and CRC code at last. We composed a malicious packet with the command we want and the suitable sequence number that is larger than the current sequence number. Using our spoofer, we could send a malicious packet and receive the corresponding ACK packet successfully. We also noticed that WCU only checks the addresses, sequence number, and CRC code. The sequence number only has to be larger than the previous sequence number. In this case, `0xFFFFFFFF` is the best choice.

We implemented an attack against DCU to masquerade power usage information. This attack was a bit more challenging than the previous attack, because we needed to send the packet exactly on time when the DCU asked WCU. In addition, our spoofing packet should be sent faster than the legitimate packet with higher signal strength. Therefore, we sent five packets in 0.1 second immediately after the receiver sniffed the power consumption information request. Finally, we could inject the power consumption information to the DCU display.

7.2 Spoofing Door Lock

In the case of door lock, capturing a real command packet is the most difficult aspect. This is because the action followed after the “open” command itself is simple. We copied the door open packet from the central controller and only a brute-force one byte sequence number field. Brute-forcing a one byte field took less than one minute, and we opened the door lock with our spoofing packets. We collected packets during brute-forcing, and found interesting result. Sequence number of packet use the value increased by one greater than the previous packet in general. However, once packet having sequence number `00` was transmitted to central controller, the sequence number initialized and increased from the value `00`. This means that we are able to open the door with just one packet having sequence number value `00`. Moreover, with address modification, we successfully opened the other rooms as desired.

8 Conclusion

In this paper, we analyzed and implemented spoofing attacks against the real world applications using unknown customized protocols upon IEEE 802.15.4 stacks such as smart plug, door lock and PSD system. We collected the packets of the target devices wirelessly, and reverse-engineered most fields of unknown customized protocols to take control of the targets by spoofing.

The results of our reverse-engineering analysis are evaluated by successful spoofing attacks against real world systems. We were able to remotely turn on/off electric devices connected to Smart plug. Moreover, we could attack the target devices more ingeniously with power usage information modification by spoofing. Likewise, opening door lock is possible with simple brute-forcing in less than one minute. The custom protocol of a PSD system, the most critical application, was also successfully analyzed. We found enough information to implement a spoofing attack with two bytes brute-forcing. For public security and safety, we could not actually conduct a test against a PSD system. However, we believe that our spoofing attack would affect the PSD system and we specifically predict that closing the PSD system would be possible.

From the results of our analysis on IEEE 802.15.4 based customized protocols, we pointed out general security problems. Wireless communication over open standard protocols can be accessed by anyone. Therefore, it must be designed with elaborate security measures.

References

1. "IEEE Standard for Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)," 2011.
2. N. Sastry and D. Wagner, "Security considerations for IEEE 802.15.4 networks," in *WiSe*. IEEE, 2004.
3. R. Sokullu, O. Dagdeviren, and I. Korkmar, "On the IEEE 802.15. 4 MAC layer attacks: GTS attack," in *SENSORCOMM*. ACM, 2008.
4. P. Jokar, N. Arianpoo, and V. C. Leung, "Spoofing detection in IEEE 802.15.4 networks based on received signal strength," *Ad Hoc Networks*, 2013.
5. ——, "Spoofing prevention using received signal strength for ZigBee-based home area networks," in *SmartGridComm*. IEEE, 2013.
6. "KillerBee," https://code.google.com/p/killerbee/.