Testing the Security ESP32 Internet of Things Devices

Oleksii Barybin, Elina Zaitseva, Volodymyr Brazhnyi
Department of Radiophysics and Cybersecurity
Vasyl' Stus Donetsk National University
Vinnytsia, Ukraine
o.barybin@donnu.edu.ua, zaytseva.elina@gmail.com, brazhnyi.v@donnu.edu.ua

***Abstract* — The physical model of a handmade IoT system that includes device for measuring temperature based on ESP32, WiFi home network and web interface was proposed and implemented upon laboratory scale. The result of the experiment based on this model to attempt to gain unauthorized access to the transmitted data was successful. Attack scenario was formulated and consist of four stages: gaining unauthorized access to a network, network traffic interception and analysis, create fake ESP32 client and disconnecting original ESP32 from a server. It is shown that the attacker, who has the basic knowledge and skills in working with common wireless network hacking tools and a basic knowledge of ESP32 and ESP32 programming skills can access the system and send fake information to the web interface. To reduce the probability of the proposed scenario it is recommended to use TCP instead of UDP.**

Keywords — Internet of Things, attack scenario, ESP32, WiFi

I. Introduction

Internet of things (IoT) technology is becoming more popular in the modern hi-tech society. Even though the precise definition of IoT is almost unachievable we can talk about IoT as an Internet connected objects that can collect data, share collected data with each other and provide the processed data to the user [1]. Due to the combination of a large number of technologies in IoT there is no structured approach to ensure appropriate information security in general, but only for specific areas and decisions: autonomous vehicle, embedded systems [2], connected vehicles, eHealth, Smart Grid [3], middleware, Smart City platform, embedded devices [4], middleware, IIoT WAN, Smart Factories [5], Sybil attacks in vehicular networks, smart home systems, smart building [6], Risk and IoT control [7], IoT in enterprise [8].

In most cases IoT systems that are produced professionally has a certain level of information security. Today, however, kits of IoT devices developed for the design and assembly at home become more common. The projects based on a microcontroller ESP32 and described at the literature [9, 10] as well as on a large number of websites gain more and more popularity. When implementing such a project for IoT device's data transmitting home wireless Wi-Fi network is used. In such system the threats for information security in the first place will be connected with using a wireless network vulnerabilities. So, finding out particular attack scenario to compromise the data transmitted from devices based on ESP32 is a topical problem.

Thus, the objective of this study is to investigate the possibility of using the vulnerabilities of wireless Wi-Fi networks to compromise data transmitted from devices based on ESP32. According to the stated objective the tasks of the study are:

- set a physical model of an ESP32 microcontroller-based IoT handmade system;
- define the initial conditions of the experiment;
- determine a possible attack scenario to compromise the data transmitted in Wi-Fi network from the device based on ESP32.

II. Physical Model

In 2015 microcontroller ESP32 was released on the market. This is an outstanding device, and not only because of its low price. The combination in a single chip Wi-Fi and Bluetooth, two core processor and a rich set of peripherals makes ESP32 leader in its segment.

Consider the basic prerequisites for the establishment of a physical model of the system that may be used for the providing experiment to compromise data transmitted from device based on ESP32.

It was demonstrated in [11] that ESP32 can be successfully used in the data acquisition and control of various devices via wireless networks and it performs much better than its predecessor ESP8266. Detailed comparative analysis ESP32 with ESP8266 can be found in [11] and the authors recommend using the microcontroller for non-professional developers. As the example of using ESP32 is the following: a user with a mobile phone can remotely check the status of his own smart home (temperature, humidity in the house), referring directly to the device.

Analysis of projects described in [9, 10] and presented on such websites as [12, 13] and others showed that one of the most popular projects of IoT devices based on ESP32 is a meteorological station. To implement this kind of project in accordance with the classical architecture of IoT (consists of three layers - application, network and interface) it is considered necessary:

- connect the sensors for measuring the parameters of environment,
- configure the network for data transmission over the home Wi-FI network,

978-1-7281-4184-8/19/$31.00 ©2019 IEEE

- implement a simple web-based interface for receiving data from the sensors.

ESP32 devKit V2 was used as base of the device. For modelling it is sufficient to connect only temperature sensor or combined temperature and humidity sensor. Manufacturers offer a large number of such sensors with approximately similar characteristics. As one of the options digital temperature sensor DS18B20 from Dallas Semiconductor can be used. The characteristics of such sensor can be found, for example, in [14].

Given the fact that it is assumed implementing home Wi-Fi network in the physical model we offer to use as an access point one of the most popular in this segment router TL-WR841N by TP-Link.

As the result the following equipment was used at the application and network level:

- small-sized ESP32-based development board ESP32 devKit V2 produced by Espressif,
- digital temperature sensor DS18B20,
- router TL-WR841N.

At the interface level to display data received from the sensor used node-red tool and, in particular, the module node-red-dashboard (Fig. 1).

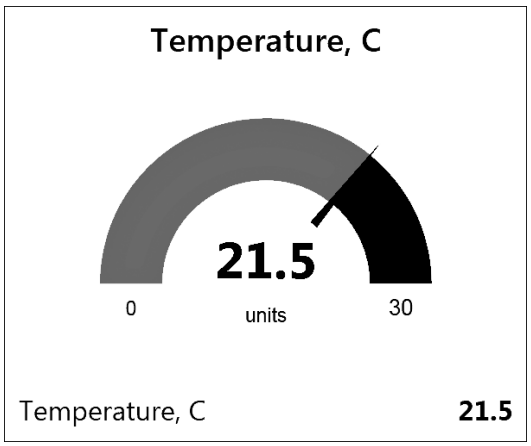

Fig. 1. Value of the temperature displayed with the module node-red-dashboard

III. Determination of the Initial Conditions of the Experiment

As already mentioned above, it is the network layer that is responsible for data security and transmission and on that layer the highest chances for implementing an attack on the device, the main element of which is the ESP32. As examples of attacks that can occur at the network layer we can mention Malicious Code Injection, Sniffing Attack, Spear-Phishing Attack, DoS (Denial-of-Service Attack), Sybil Attack, Proxy Attack, Sleep Deprivation attack, etc [1].

When transferring data from the IoT devices different protocols can be used, but as shown in [15, 16] all of them based on TCP or UDP. In our case we use the UDP protocol. It uses a simple data model and is most often used for systems with sensors because of communication overhead reducing. The negative features of this protocol are that it does not guarantee delivery or proper sequence of datagrams and does not provide connection and resending services in case of data loss.

Mechanisms of Wi-Fi networks protection include authentication (the client and the access point are presented to each other and confirm the right to exchange data) and encryption (selection of the data encryption algorithm and keys generation). As stated in [17] in home WiFi networks it is recommended to use the authentication and data encryption based on WPA2-PSK specification.

The vulnerabilities of Wi-Fi networks security protocols are described, for example, in [18]. WPA and WPA2 protocols encrypt data separately for each client with a temporary key that generated after a client connects to the access point. To get the key you need to know the parameters of the network, which can be freely intercept by network traffic eavesdropping, and the main obstacle to an attacker is to obtain Pre-Shared Key by trying all possible password combinations or using a dictionary. To attempt password guessing attacker must intercept the handshake between the client and the access point. Password matching rate depends on the speed of the attacker's computer and password dictionary capacity.

There are many tools that can be used by an attacker to exploit the WiFi network vulnerabilities. The most commonly used are the tools stated below.

Aircrack-ng is an 802.11 WEP and WPA-PSK keys cracking program that can recover keys once enough data packets have been captured. It implements the standard FMS attack along with some optimizations like KoreK attacks, as well as the all-new PTW attack, thus making the attack much faster compared to other WEP cracking tools. Additionally, the program offers a dictionary method for determining the WEP key. For cracking WPA/WPA2 pre-shared keys, only a dictionary method is used [19].

Airmon-ng is included in the aircrack-ng package and is used to enable and disable monitor mode on wireless interfaces. It may also be used to go back from monitor mode to managed mode [19].

Airodump-ng is included in the aircrack-ng package and is used for packet capturing of raw 802.11 frames. It is ideal for collecting WEP IVs for use with aircrack-ng [19].

Aireplay-ng is included in the aircrack-ng package and is used to inject wireless frames. Its main role is to generate traffic for later use in aircrack-ng for cracking WEP and WPA-PSK keys. Aireplay-ng has many attacks that can deauthenticate wireless clients for the purpose of capturing WPA handshake data, fake authentications, interactive packet replay, hand-crafted ARP request injection, and ARP-request reinjection [19].

Besside-ng is a tool that support also WPA encryption. It will crack automatically all the WEP networks in range and log the WPA handshakes. Requirements: Aircrack-ng SVN version, Wireless interface with working injection [19].

Wireshark is the world's foremost network protocol analyzer. Wireshark has a rich feature set which includes the following: deep inspection of hundreds of protocols, with more being added all the time, live capture and offline analysis, standard three-pane packet browser, captured network data can be browsed via a GUI, or via the TTY-mode TShark utility, decryption support for many

protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL / TLS, WEP, and WPA / WPA2 [19].

All of these tools can be found in the widely available special operating systems builds based on the Linux operating system: Kali Linux, Parrot Security OS, Black Arch etc.

Thus, the initial conditions of the experiment include the following:

- the use of the UDP protocol,
- authentication and data encryption based on WPA2-PSK specification,
- attacker skill level sufficient for use Aircrack-ng tools, Airmon-ng, Airodump-ng, Aireplay-ng, Besside-ng, Wireshark.

The intended result of the experiment is to substitute temperature data from the sensor in the user device by the fake temperature data generated by the prospective attacker.

IV. Experimental Results and Discussion

In the experiment Parrot Security OS was used. The name of the attacker computer's network card, through which will be carried out unauthorized access, defined as wlan1. The name of the Wi-Fi network, which transmits the temperature data from the sensor to web interface is STAARS. To obtain the password to the network we used a dictionary of passwords from rockyou developers, which is a universal dictionary to access a Wi-Fi network.

At the first stage, we need to get unauthorized access to STAARS. To do this, perform the following steps:

- terminate all the processes connected to the network (command airmon-ng check kill),
- change the wlan1 mode to Monitor Mode (command airmon-ng start wlan1),
- implement preview of all available access points to select the desired,
- create another terminal to continue to monitor the network packets,
- capture handshake (beside-ng), which is automatically recorded in wpa.cap file.
- run password guessing process.

At the second stage, we need to intercept and analyze network traffic. To do this, we use Wireshark (Fig. 2).

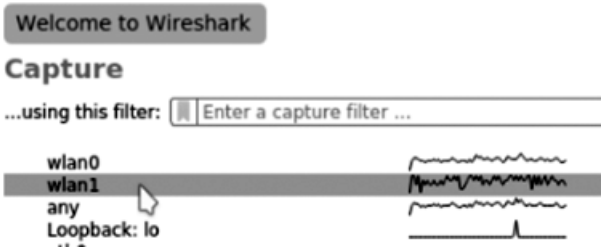

Fig. 2. Choosing the network in Wireshark

Fig. 3 shows the information necessary to disconnect the client from the server and to connect attacker`s computer to the server, which transmits the temperature data and create a fake client that sends non-original data.

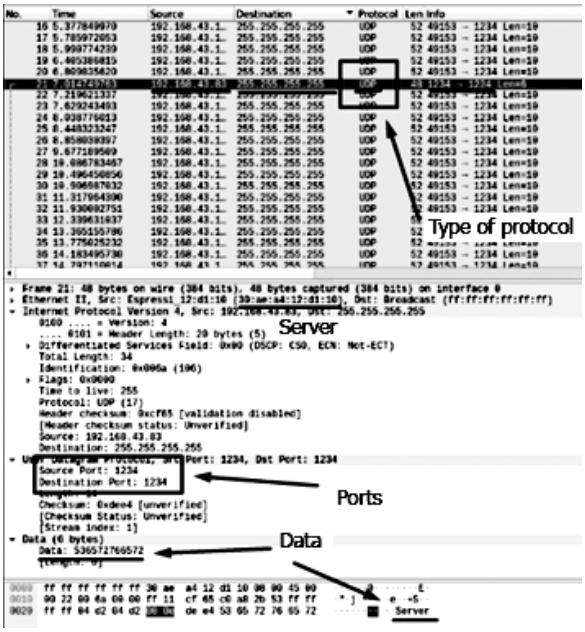

Fig. 3. Wireshark interface screenshot specifying desired data.

At the third stage, we use Arduino IDE to create fake client. Fig. 4 shows the code that uses the information gathered in Wireshark.

Fig. 4. Programming code for fake client.

At the fourth stage after starting the fake client, we disconnect original client from the server (Fig. 5).

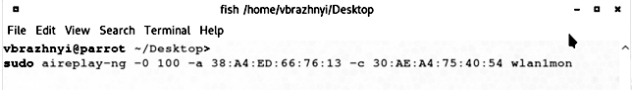

Fig. 5. Disconnecting the original client from the server.

Results of the STAARS network traffic analysis (Fig. 5) show that the fake data are transmitted to the network.

Fig. 6. STAARS network traffic

Thus, as the result the following attack scenario can be formulated:

- Gaining unauthorized access to a network (network card transfers to monitor mode (Airmon-ng), view available access points (Airodump-ng), handshake interception, guessing the password (Besside-ng).
- Network traffic interception and analysis (Wireshark).
- Create fake ESP32 client using the captured data (Arduino) and connect it to the network.
- Disconnecting original ESP32 from a server (Aircrack-ng).

As seen from the described scenario for the first, second and fourth stages attacker's qualification must be sufficient to use a standard hacking tools for working with wireless networks. Actions in phase 3 presuppose basic skills in programming ESP32. Overall, however, the described scenario can be implemented by an attacker with a low level of knowledge and skills in networking and the ESP32 programming. This allows to implement the above-mentioned scenario easy enough.

It should be noted that in the implemented network configuration most vulnerable point is UDP protocol. It does not require checking the status of the sent data packets. This allows attacker to intercept and replace data packets. Accordingly, in order to prevent such a scenario, it is sufficient to use the TCP protocol, which is in contrast to UDP ensures data integrity and sender notification of transmission results.

V. Conclusion

Analysis of current trends in IoT technologies enabled us to identify a segment for which information security assurance may encounter a lack of its level. These are IoT devices based on ESP32 microcontroller, that designed and implemented at home by non-professionals. In this paper we justified the technical solutions to create a physical model of a simple information collection and transmission system that can transmit environmental temperature data to the user's web interface. Implemented model was used in carrying out the experiment under laboratory conditions to attempt to gain unauthorized access to the transmitted data.

In the result of the experiment attack scenario was described. It is shown that the attacker, who has the basic knowledge and skills in working with common wireless network hacking tools (Aircrack-ng, Airmon-ng, Airodump-ng, Aireplay-ng, Besside -ng, Wireshark) and a basic knowledge of ESP32 and ESP32 programming skills can access the system and send fake data to the web interface. To reduce the probability of the proposed scenario it is recommended to use TCP instead of UDP.

References

[1] A. Colakovic and M. Hadžialic "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues", *Computer Networks*, vol. 144, pp. 17-39, 2018. DOI: https://doi.org/10.1016/j.comnet.2018.07.017

[2] Pathan A.K. *Securing Cyber-Physical Systems*. Boca Raton:CRC Press, 2015. DOI: https://doi.org/10.1201/b19311

[3] Misra S., Maheswaran M. and Hashmi S. *Security Challenges and Approaches in Internet of Things*. Springer, 2017. DOI: https://doi.org/10.1007/978-3-319-44230-3

[4] Aziz B., Arenas A. and Crispo B. *Engineering Secure Internet of Things Systems*. London:CPI Group, 2016. DOI: https://doi.org/10.1049/PBSE002E

[5] Gilchrist A. *Industry 4.0. The Industrial Internet Of Things*. Apress, 2016. DOI: https://doi.org/10.1007/978-1-4842-2047-4

[6] Hu F. *Security and Privacy in Internet of Things*. Boca Raton:CRC Press, 2016. DOI: https://doi.org/10.1016/B978-0-12-805395-9.00010-1

[7] Macaulay T. *RIoT Control. Understanding and Managing Risks and the Internet of Things*, Cambridge:Elsevier, 2017. DOI: https://doi.org/10.1016/B978-0-12-419971-2.00001-7

[8] Russell B. and Duren D. *Practical Internet of Things Security*. Birmingham:Packt Pub, 2016.

[9] Ibrahim D. *The Complete ESP32 Projects Guide. 59 Experiments with Adruino IDE and Python*. Elektor, 2019.

[10] Kurniawan A. *Internet of Things Projects with ESP32. Build exciting and powerful IoT projects using the all-new Espressif ESP32*. Birmingham:Packt Pub, 2019.

[11] A. Maier, A. Sharp, Y. Vagapov "Comparative Analysis and Practical Implementation of the ESP32 Microcontroller Module for the Internet of Things", *in 2017 Internet Technologies and Applications (ITA)*, Wales, UK, 2017, pp. 1-6. DOI: https://doi.org/10.1109/ITECHA.2017.8101926

[12] "20+ ESP32 Projects and Tutorials | Random Nerd Tutorials", *Random Nerd Tutorials*, 2019. [Online]. Available: https://randomnerdtutorials.com/projects-esp32/. [Accessed: 12- Aug- 2019].

[13] "The Internet of Things with ESP32", *Esp32.net*, 2019. [Online]. Available: http://esp32.net/. [Accessed: 12- Aug- 2019].

[14] *DS18B20 Programmable Resolution 1-Wire Digital Thermometer*. Maxim Integrated, 2018.

[15] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, J. Alonso-Zarate "A Survey on Application Layer Protocols for the Internet of Things", *Transaction on IoT and Cloud Computing*, vol. 1, no. 1, p. 1-10, 2015. DOI: https://doi.org/10.5281/zenodo.51613

[16] T. Salman, R. Jain "A Survey of Protocols and Standards for Internet of Things", *Advanced Computing and Communications*, Vol. 1, No. 1, p. 1-20, 2017.

[17] D. Costinela-Luminita "Wireless LAN Security - WPA2-PSK Case Study", *in 2nd World Conference on Information Technology (WCIT-2011)*, Antalya, Turkey, 2011, pp. 62-67.

[18] Salmon A., Levesque W. and McLafferty M. *Applied Network Security*. Birmingham:Packt Pub, 2017.

[19] *Tools.kali.org*, 2019. [Online]. Available: https://tools.kali.org/tools-listing. [Accessed: 12- Aug- 2019].