Security Analysis of UDP-Based Data Transmission Between ESP32 Devices over Wi-Fi: Vulnerabilities and the Need for Encryption

SABAH, Ilyasse
i.sabah@usms.ma

AIT IJJA, Abdelilah
aitijja@usms.ma

April 4, 2025

Abstract

This paper examines the security of UDP-based data transmission between two ESP32 devices over a Wi-Fi Wireless network. In the experiment, one ESP32 functions as a server (receiver), and the other as a client (transmitter). Our method demonstrates that an attacker after gaining access to the network can successfully craft and send malicious data packets to the ESP32 server. The methodology used in this study involves gaining access to the network and analyzing network packets to identify the ESP32 server. Once the server is identified, we use the python Scapy tool to generate malicious data packets and inject them into the network affecting the ESP32 server communication. The results highlight the significant security risks of using UDP for data transmission in IoT systems. We conclude that secure alternatives, such as encrypted communication protocols, should be employed to ensure the confidentiality and integrity of transmitted data, rather than relying on UDP.

Keywords: ESP32, UDP, IoT, Wi-Fi, security, encryption, data transmission

1 Introduction

The Internet of Things (IoT) represents a revolutionary advancement in the way the world connects and interacts with technology. By enabling everyday objects to communicate and share data through the internet, IoT has brought about a profound transformation across various sectors, including healthcare, agriculture, transportation, manufacturing, and smart homes.

IoT devices range from simple sensors that monitor environmental conditions to complex systems that control entire infrastructures. These devices are capable of transmitting real-time data, enabling immediate responses and informed decision-making, which is essential for improving efficiency, convenience, and security.

The ESP32, a low-cost, low-power microcontroller with integrated Wi-Fi and Bluetooth capabilities, has become one of the most popular platforms for IoT applications. It is widely used in various IoT projects due to its versatility, compact size, and robust performance. The ESP32 is capable of connecting to Wi-Fi networks, enabling communication with other devices or servers, and it can also perform complex tasks, such as sensor data processing, real-time communication, and even controlling other devices. Its versatility makes it ideal for a wide range of IoT applications, from home automation systems and wearables to environmental monitoring and industrial automation.

The power of the ESP32 in IoT systems lies in its ability to enable both local device-to-device communication and remote cloud-based integration. With its built-in wireless capabilities, it allows for seamless integration into larger IoT networks and supports real-time data transfer, which is essential for monitoring and control in connected environments. Additionally, the ease of use and accessibility of development tools for the ESP32, such as the Arduino IDE, make it a popular choice for both beginners and experienced engineers alike.

In this paper, we explore the security implications of UDP-based communication in IoT networks, focusing on the ESP32 devices used in this study. We highlight the potential risks of relying on insecure communication protocols and recommend strategies for enhancing data security in IoT systems.

2 Methodology

The focus is on demonstrating how vulnerabilities in UDP-based communication can be exploited in an IoT environment. The experiment involves three key phases: gaining network access, packets capture and analysis, and generation and injection of malicious packets.

2.1 Gaining Network Access

Wi-Fi networks typically rely on encryption protocols, such as WEP, WPA, WPA2, to secure data transmissions between devices. While these protocols offer a level of security, they are not foolproof. Over time, various weaknesses in encryption mechanisms have been discovered, allowing attackers to potentially intercept, eavesdrop, or even gain unauthorized access to a network. In such cases, attackers can exploit these weaknesses to launch targeted attacks on the network.

Hackers typically use two main techniques to gain unauthorized access to wireless networks: Packet sniffing and dictionary attacks, or Wi-Fi phishing (Evil Twin attacks).

1. **Packet Sniffing and Dictionary Attacks:** In this method, hackers monitor wireless traffic using packet-sniffing tools such as Wireshark or Kismet to capture unencrypted data or the WPA/WPA2 handshake. Once the handshake is captured, they use a dictionary or wordlist (a precompiled list of likely passwords) in combination with tools like Aircrack-ng to guess the password. This is especially effective against networks that use weak or common passwords.

2. **Wi-Fi Phishing (Evil Twin Attacks):** An attacker sets up a fake access point with the same SSID (network name) as a legitimate network, often in public places, and tricks users into connecting to it. This "Evil Twin" access point allows the attacker to intercept data, monitor traffic, and potentially steal sensitive information like login credentials. Tools like Fluxion or Karma can automate the process of impersonating a legitimate network and capturing user data.

These techniques exploit various weaknesses in wireless security protocols, and they are commonly used by hackers to gain unauthorized access to Wi-Fi networks for malicious purposes.

For our experiment, we created a controlled testing environment using a commercial router and two ESP32 devices. The commercial router provided a stable and secure network infrastructure for the devices to communicate over, ensuring that the experiment closely mimicked real-world IoT network scenarios. The ESP32 devices were chosen for their versatility and widespread use in IoT applications, with one ESP32 configured as client, transmitting UDP packets, and the other as the server, receiving and processing the packets. This setup allowed us to simulate typical IoT communication while maintaining full control over the network and device configurations, making it an ideal environment to study the vulnerabilities of UDP-based communication in IoT systems.

In our experiement we have used :

1. ZTE ZXHN F680 (to establish the Wi-Fi network).
2. Two ESP32 WROOM-32 modules (functioning as the communicating IoT devices).

Figure 1: ZTE ZXHN F680

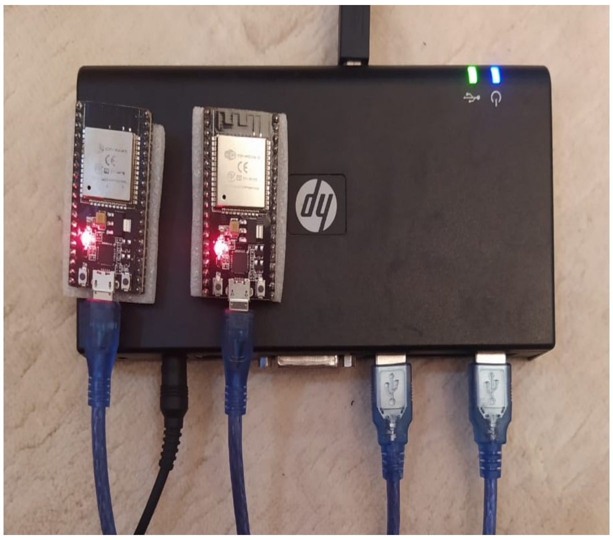


Figure 2: ESP32 Client and Server setup

The server is programmed to continuously wait for incoming UDP packets from the client, process the received data, and then display it in a readable format. This server-client setup allows us to simulate real-time data exchange in an IoT environment, where the server acts as the receiver and the client sends relevant information. Below is an example of the Serial Output from both the ESP32 server and client, demonstrating how the data exchange takes place between the two devices:

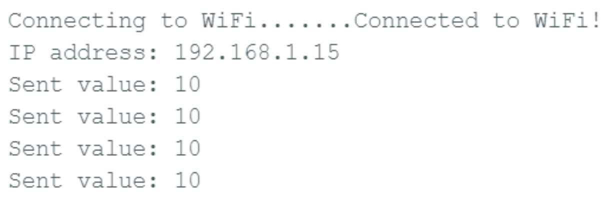

```
Connecting to WiFi.......Connected to WiFi!
IP address: 192.168.1.15
Sent value: 10
Sent value: 10
Sent value: 10
Sent value: 10
```

Figure 3: Serial output from ESP32 Client (sender)

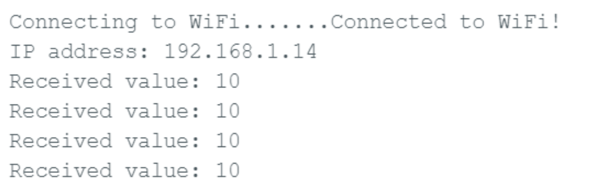
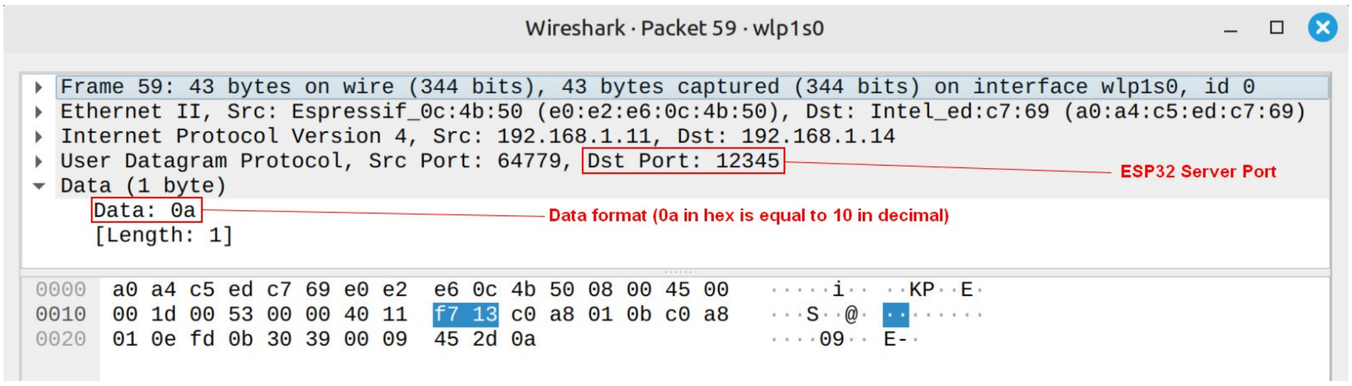

```
Connecting to WiFi.......Connected to WiFi!
IP address: 192.168.1.14
Received value: 10
Received value: 10
Received value: 10
Received value: 10
```

Figure 4: Serial output from ESP32 Server (receiver)

2.2 Packets capture and analysis

After successfully gaining access to the network, we proceeded to scan for available devices. Upon identifying the target, we conducted a Man-in-the-Middle (MITM) attack on all devices using Bettercap. This allowed us to intercept and capture the UDP data packets, enabling us to analyze the communication protocol using Wireshark and gain insights about the data used between the two ESP32 (ESP32 Server IP, Service port , Data format).


Wireshark · Packet 59 · wlp1s0
ESP32 Server Port
Data format (0a in hex is equal to 10 in decimal)

```
▸ Frame 59: 43 bytes on wire (344 bits), 43 bytes captured (344 bits) on interface wlp1s0, id 0
▸ Ethernet II, Src: Espressif_0c:4b:50 (e0:e2:e6:0c:4b:50), Dst: Intel_ed:c7:69 (a0:a4:c5:ed:c7:69)
▸ Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.14
▸ User Datagram Protocol, Src Port: 64779, Dst Port: 12345
▾ Data (1 byte)
    Data: 0a
    [Length: 1]

0000  a0 a4 c5 ed c7 69 e0 e2  e6 0c 4b 50 08 00 45 00   ·····i·· ··KP··E·
0010  00 1d 00 53 00 00 40 11  f7 13 c0 a8 01 0b c0 a8   ···S··@· ········
0020  01 0e fd 0b 30 39 00 09  45 2d 0a                  ····09·· E-·
```

Figure 5: UDP packet format from Wireshark

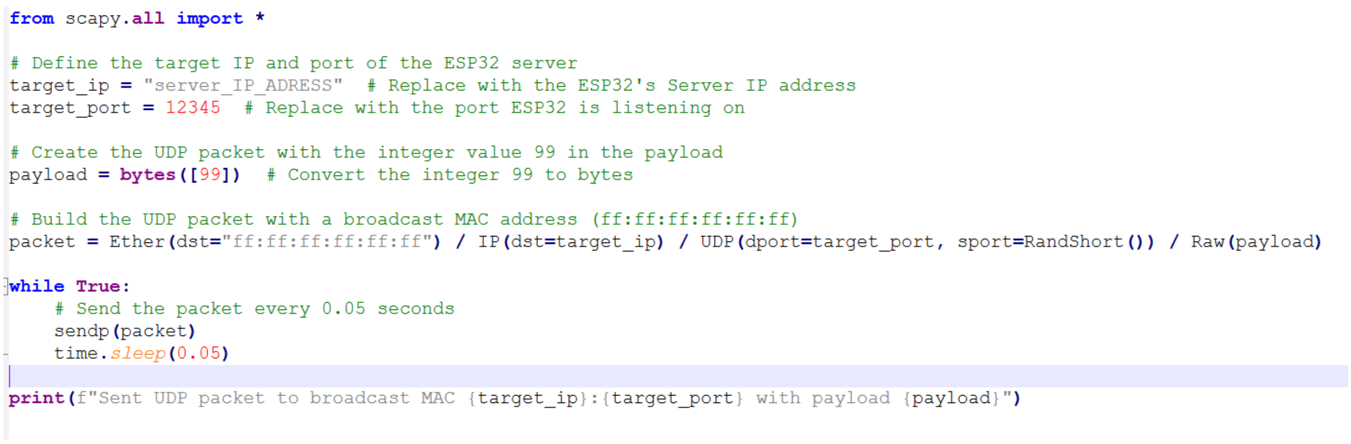
2.3 Generation and injection of malicious packets

Scapy is a powerful Python-based interactive tool that is used for network packet manipulation, crafting, and analysis. It allows users to interact with network protocols, create custom packets, send them over a network, and analyze responses.

Scapy is widely used by cybersecurity professionals, network administrators, and penetration testers due to its versatility and ease of use.

Based on the informations above, we leveraged Scapy to craft a custom UDP packet that precisely matched the observed data structure.

By specifying the correct port and replicating the data format, we were able to create a malicious packets designed to exploit the vulnerabily in the UDP protocol communication.

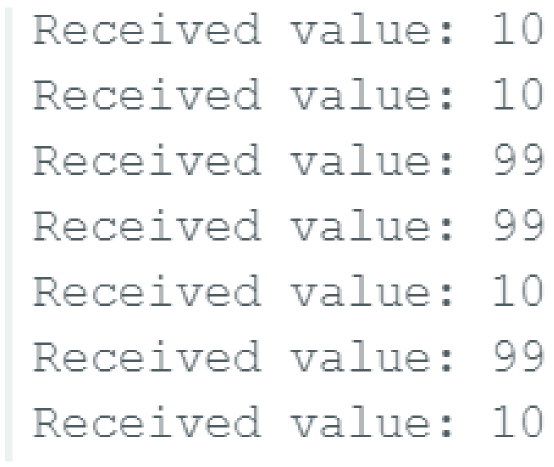


```python
from scapy.all import *

# Define the target IP and port of the ESP32 server
target_ip = "server_IP_ADRESS"  # Replace with the ESP32's Server IP address
target_port = 12345  # Replace with the port ESP32 is listening on

# Create the UDP packet with the integer value 99 in the payload
payload = bytes([99])  # Convert the integer 99 to bytes

# Build the UDP packet with a broadcast MAC address (ff:ff:ff:ff:ff:ff)
packet = Ether(dst="ff:ff:ff:ff:ff:ff") / IP(dst=target_ip) / UDP(dport=target_port, sport=RandShort()) / Raw(payload)

while True:
    # Send the packet every 0.05 seconds
    sendp(packet)
    time.sleep(0.05)

print(f"Sent UDP packet to broadcast MAC {target_ip}:{target_port} with payload {payload}")
```

Figure 6: Python based code to craft malicious packets and inject them into the network (Scapy library)

3 Results

After the injection of the UDP packets into the network, and due to the security weakness in the UDP Protocol, The ESP32 Server processes the injected data without any validation. The results of these interactions (Real data from legitimate ESP32 Client and Malicious packets injected into the network) are displayed below, showing how the ESP32 server reacts to an unintended or unexpected manner.



```
Received value: 10
Received value: 10
Received value: 99
Received value: 99
Received value: 10
Received value: 99
Received value: 10
```

Figure 7: Serial output from ESP32 Server (After Packets injections)

4 Conclusion

The results of our research highlight several key vulnerabilities within the UDP (User Datagram Protocol) that pose significant risks, particularly for devices like the ESP32 which rely on UDP for lightweight, low-latency communication over Wi-Fi networks

While prior research in the field of UDP vulnerability exploits often relies on additional devices or network disruptions (e.g., deauthentication), our approach diverges significantly from these methods in several critical aspects. Unlike previous research, our technique does not require a second ESP32 device or the deauthentication of the target server. This simplification allows for a more efficient and accessible method to test UDP vulnerabilities without relying on additional hardware or needing to interfere with legitimate traffic.

By using a single PC and the powerful capabilities of its CPU, we can perform UDP packet injection on one or multiples ESP32 Server with a high rate of requests, allowing for flood testing and server overload. This technique can be easily replicated using tools like the Python Scapy library, which is lightweight, flexible, and doesn't require specialized hardware.

The ability to flood a network using just a PC opens up new possibilities for testing the resilience of IoT devices under stress, as it eliminates the need for multiple attacking devices or complex setup procedures. With the ability to inject packets at a high rate, our approach presents a scalable and straightforward method for evaluating UDP vulnerabilities on devices like the ESP32.

However, it is crucial to note that the vulnerabilities we identified in UDP can lead to significant security risks, particularly in sensitive applications. To mitigate these risks, we strongly recommend implementing encryption and end-to-end verification mechanisms. Encryption ensures that the data transmitted between devices remains confidential and resistant to interception, while end-to-end verification allows devices to authenticate each packet, ensuring the integrity and authenticity of the communication

References

[1] Oleksii Barybin, Elina Zaitseva, and Volodymyr Brazhnyi (2019). "Testing the Security of ESP32 Internet of Things Devices". IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC ST), Kyiv, Ukraine, 2019, pp. 143-146, doi: 10.1109/PICST47496.2019.9061269.

[2] Guo, Rui. "Survey on WiFi infrastructure attacks." Int. J. Wirel. Mob. Comput. 16 (2019): 97-101, doi: 10.1504/IJWMC.2019.099026.

[3] Jenish Rudani, Daksh Khorana, Yagnesh Savaliya "Cryptographic Communication between Two ESP32 Devices" International Research Journal of Engineering and Technology (IRJET),Volume: 08 Issue: 01 Jan 2021,p-ISSN: 2395-0072,e-ISSN: 2395-0056.

[4] R. R. S, R. R, M. Moharir and S. G, "SCAPY- A powerful interactive packet manipulation program," 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS), Bangalore, India, 2018, pp. 1-5, doi: 10.1109/ICNEWS.2018.8903954.

[5] Mohammad Al-Mashhadani,Mohamed Shujaa, "IoT Security Using AES Encryption Technology based ESP32 Platform", The International Arab Journal of Information Technology (IAJIT) ,Volume 19, Number 02, pp. 214 - 223, March 2022, doi: 10.34028/iajit/19/2/8.