J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

A Study on Vulnerability Analysis and Memory Forensics of ESP32☆

Jiyeon Baek[1] Jiwon Jang[1] Seongmin Kim[2*]

ABSTRACT

As the Internet of Things (IoT) has gained significant prominence in our daily lives, most IoT devices rely on over-the-air technology to automatically update firmware or software remotely via the network connection to relieve the burden of manual updates by users. And preserving security for OTA interface is one of the main requirements to defend against potential threats. This paper presents a simulation of an attack scenario on the commoditized System-on-a-chip, ESP32 chip, utilized for drones during their OTA update process. We demonstrate three types of attacks, WiFi cracking, ARP spoofing, and TCP SYN flooding techniques and postpone the OTA update procedure on an ESP32 Drone. As in this scenario, unpatched IoT devices can be vulnerable to a variety of potential threats. Additionally, we review the chip to obtain traces of attacks from a forensics perspective and acquire memory forensic artifacts to indicate the SYN flooding attack.

☞ keyword : Memory forensics, Over-The-Air (OTA), ESP32, Attack scenario

1. Introduction

Despite the widespread adoption of IoT technology, preserving security guarantees for IoT devices has been less of a priority than functionality and performance due to the limited hardware resources and compact size, making developers focus on delivering concise features. This tendency, in turn, makes designing and implementing robust security in IoT devices challenging and leads to various security incidents. For example, Mozi botnet specifically targeted IoT devices for use in Distributed Denial of Service (DDoS) attacks. It successfully infected 12,000 IoT devices across 72 countries [1]. Additionally, wall-pad IoT devices were hacked in South Korea, resulting in privacy leakage for over 400,000 households through the device's built-in camera in 2021 [2].

Meanwhile, most IoT devices rely on over-the-air (OTA) technology to automatically update firmware or software remotely via the network connection to relieve users of the burden of manual updates. By utilizing OTA interfaces, devices can seamlessly receive and install updates over a network connection, ensuring that they stay up to date with the latest features, bug fixes, and security patches. This technology is best known for updating smart cars, but due to its convenience, OTA interfaces are also widely used in smartphones and smart home devices. Moreover, with the increasing adoption of modern system-on-a-chip (SoC) designs in embedded systems and IoT devices, wireless connectivity features such as WiFi and Bluetooth Low Energy (BLE) are becoming commonplace. Many of these SoCs are equipped with OTA update capabilities, further increasing the convenience and prevalence of OTA.

However, despite of the ubiquity of OTA technology, potential security vulnerabilities still exist when utilizing OTA. Attacker can interpose the OTA channel by sniffing the update packets, taking unauthorized device control, and installing malware [3]. Once the OTA connection is compromised, an attacker gains the ability to update the

1 Dept. of Future Convergence Technology Engineering, Sungshin Women's University, Seoul, 02844, Korea
2 Dept. of Convergence Security Engineering, Sungshin Women's University, Seoul, 02844, Korea
* Corresponding author (sm.kim@sungshin.ac.kr)
[Received 27 January 2024, Reviewed 20 February 2024(R2 09 April 2024), Accepted 17 April 2024]
☆ This work is supported by the Ministry of Trade, Industry and Energy (MOTIE) under Training Industrial Security Specialist for High-Tech Industry (RS-2024-00415520) supervised by the Korea Institute for Advancement of Technology (KIAT), and the Ministry of Science and ICT (MSIT) under the ICAN (ICT Challenge and Advanced Network of HRD) program (No. IITP-2022-RS-2022-00156310) supervised by the Institute of Information & Communication Technology Planning & Evaluation (IITP).
☆ A preliminary version of this paper was presented at APIC-IST 2023.

firmware and take full control of the target device.

In this study, the security analysis on OTA updates is demonstrated using a commodity SoC called ESP32, developed by Espressif [4]. The ESP32 is a typical low-cost SoC that offers low-power consumption and rich integrations with the OTA feature [5]. It is widely used across various domains, from Arduino drones to commercial low-end wearable devices and smart home IoT. Specifically, this research focuses on an IoT drone (referred to as ESP32 drone), which is based on the ESP32 DOIT DEVKIT board, to identify potential attack scenarios during OTA updates. The contributions of this paper are outlined as follows:

(1) Implementation of attack scenarios on the OTA process for security evaluation of ESP32 drones, ultimately disrupting the firmware updates of the drones. This highlights potential vulnerabilities in unpatched firmware.

(2) Conducting a forensic analysis of the chip after the attacks to obtain traces of the attacks, demonstrating the potential for extracting hacking evidence during the OTA process.

Section 2 of the paper presents several studies related to attack scenarios and forensics in IoT. Section 3 introduces and implements attack scenarios on OTA, and Section 4 conducts forensic analysis on the ESP32 memory after the attacks. Section 5 concludes the paper.

2. Related Work

This section encompasses research studies focused on hacking pertaining to networks or OTA methods targeting IoT devices, as well as investigations conducted in the field of digital forensics concerning such devices.

Jeon and Lee [3] analyzed OTA update vulnerabilities in IoT healthcare devices constructed of Arduino MKR1000 WiFi board. They implemented reverse engineering of the sniffed OTA packet data and performed a mock attack on the OTA process to install a dummy program on the device.

Barybin et al. [6] employed various network hacking tools to simulate a hack on a digital temperature sensor built with the ESP DevKit V2. They disconnected the device's WiFi connection from the server and transmitted fake temperature data, pretending to be a victim. The study identified the UDP protocol as the most vulnerable point in this experiment and recommended using the TCP protocol for better security.

Li et al. [7] conducted practical memory forensic experiments on the ESP series to retrieve forensic evidence. They found that retrieving memory data through the USB port could be dangerous as the device automatically runs when connected, allowing potential tampering if a malicious program is installed. Instead, they classified the ESP series into three types by pins and suggested a forensic method using a combination of 3D printing, PoGo pins, and cold soldering.

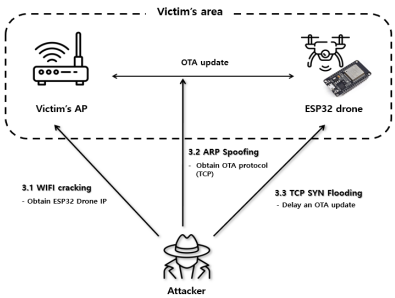
This study stands out by presenting an attack scenario specifically focused on disrupting the OTA process of the ESP32 device. In contrast, reference [3] did not specifically target the ESP32 device, and [6] exploits the WiFi connection instead of OTA. Furthermore, this paper conducts an analysis of memory forensic artifacts resulting from the attacks through UART. It is important to note that the simulated attack scenario in this study does not involve the installation of malware. As a result, it does not address the issue of memory data extraction through USB, which was discussed in [7].

3. OTA attack simulation scenario

This section demonstrates a simulation of an OTA attack scenario, as shown in Figure 1. The scenario involves a standard OTA update on an ESP32 drone initiated by a victim, alongside a simulated network attack occurring concurrently with the OTA update process. The simulation includes several steps outlined below:

- In order to gain access to a victim's network, an attacker attempts to crack the victim's WiFi network.
- After access the victim's network, the attacker identifies the IP address of the ESP32 drone.
- When the victim conducts an OTA update on an ESP32 drone, the attacker sniffs the OTA packets by performing ARP spoofing.
- The attacker draws the OTA protocol and executes a TCP SYN flooding attack while a new OTA is being conducted on the ESP32 drone to disrupt the update process.

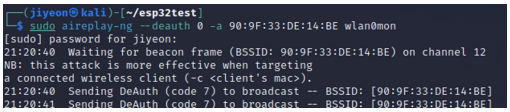
The implementation of this scenario involves using Kali Linux on the attacker's PC and utilizing an IPTIME N604R plus router as the victim's access point, to which the ESP32 drone is connected.



(Figure 1) The attack scenario on OTA process of the ESP32 drone.

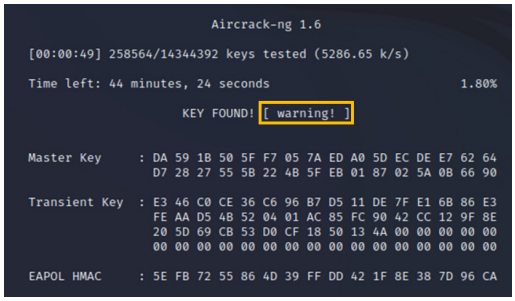
3.1 Cracking Wireless Access Point

In the initial stage of the attack scenario, several network hacking tools such as aircrack-ng, airmon-ng, airodump-ng, and aireplay-ng, known for their ability to crack WEP/WPA networks, were utilized to gain access to the victim's router [8]. The first step involved switching the wireless interface from managed mode to monitor mode using airmon-ng. This enabled the scanning of all WiFi connections, including the victim's network, using airodump-ng.



(Figure 2) Using airplay-ng to infiltrate WiFi network.

After scanning the victim's WiFi connection, aireplay-ng is employed to send de-authentication packets to the network, as depicted in Figure 2, in order to capture the WPA handshake of the victim's WiFi. Once the WPA handshake information is obtained, aircrack-ng is utilized in conjunction with the password dictionary rockyou.txt [9] to crack the password of the victim's WiFi, as shown in Figure 3. Subsequently, upon gaining access to the victim's WiFi connection, the IP address of the ESP32 drone is identified, which was utilized for ARP spoofing during the OTA update.



(Figure 3) Crack the victim's WiFi password with aircrack-ng and rockyou.txt

3.2 ARP Spoofing attack during OTA update

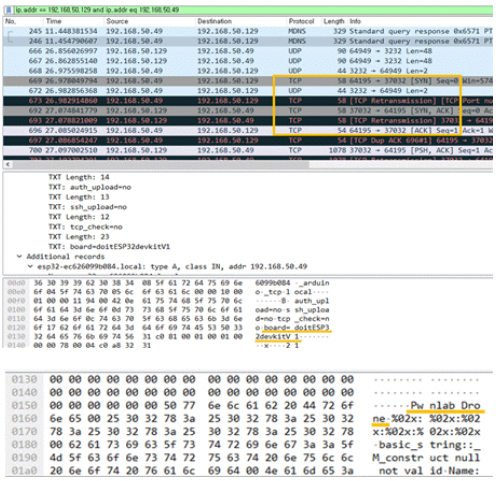
The simulation of OTA packet sniffing is conducted using Ettercap and Wireshark on the Kali Linux. Ettercap [10], an open-source tool, is utilized to facilitate man-in-the-middle (MITM) attacks on the local area network (LAN), while Wireshark [11], another open-source tool, is employed for comprehensive network packet and protocol analysis. In particular, the ARP poisoning technique, which is one of the MITM attack techniques supported by Ettercap, is employed to intercept and manipulate network traffic, allowing for the capture of OTA packets. Concurrently, Wireshark is utilized to capture and analyze the intercepted OTA packet.

Before initiating the ARP spoofing, we identified the IP addresses of both the ESP32 drone and the attacker by sniffing the packets (Figure 4).

Node	IP	MAC
ESP32 Drone	192.168.50.49	EC:62:60:99:B0:84
Attacker	192.168.50.17	90:9F:33:0B:6A:FE

(Figure 4) Deriving IP and MAC addresses

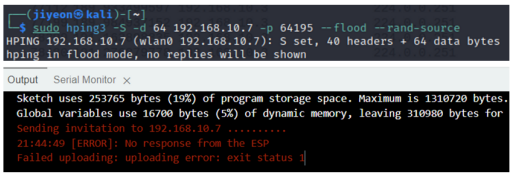
During the ARP spoofing process, it was verified that the ESP32 drone, which had previously established a connection to the network using the UDP protocol, initiated an OTA update and established a TCP connection for the transmission of the update data. Moreover, Figure 5 illustrates additional details obtained during this process, including information such as the board type of ESP32 and user-designated strings.



(Figure 5) Discovered TCP 3-way handshake connection and ESP32 drone information during OTA update

3.3 TCP SYN Flooding attack during OTA update

Upon discovering that the OTA update utilizes TCP protocol, a TCP SYN flooding attack is executed to disrupt the OTA process on the drone. TCP SYN flooding is a form of Denial-of-Service(DoS) attack that exploits the TCP 3-way handshake of a target. To carry out this attack, the hping3 tool [12] is employed to repeatedly transmit SYN packets to the ESP32 drone. As a result, the update server ceases its attempts to establish a connection with the drone and generates an error message. Figure 6 illustrates the command used for the attack and the error message displayed from the server.



(Figure 6) The OTA update process fails when a TCP SYN packet is sent to the ESP32 drone.

In summary, the attack scenario involved three attacks on the ESP32 drone during its OTA update process. Initially, the attacker gained unauthorized access to the access point to which the ESP32 drone was connected, by exploiting WiFi cracking techniques. Subsequently, an ARP spoofing attack was executed, revealing that the ESP32 OTA process utilized the TCP protocol as the default. Finally, TCP SYN flooding attack was launched to disrupt the update process, resulting in a delay in patching the ESP32 drone with the latest firmware.

Maintaining the most up-to-date version of software and promptly patching vulnerabilities through software updates is crucial for ensuring device security. This OTA attack scenario highlighted that if an attacker continues to hinder OTA updates with malicious intent, the vulnerable software version may persist, leaving the possibility of future hacking.

4. ESP32 Memory Analysis

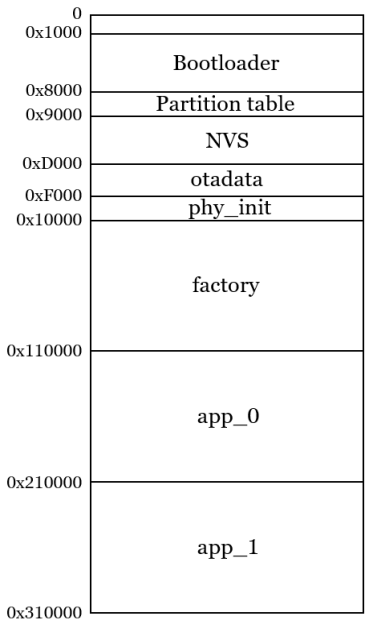
To excavate the forensic evidence of TCP SYN flooding attack aimed at delaying OTA updates, an analysis of the ESP32-WROOM-32 is conducted, which is the fundamental component of the ESP32 DOIT DEVKIT board. This particular module comprises 520KB of internal SRAM, 448KB of internal ROM, and 4MB of external SPI (Serial Peripheral Interface) flash storage which is non-volatile and stores user data [13]. The connection is established using the Universal Asynchronous Receiver /Transmitter (UART) interface, while the memory dump is performed using the esptool.py program, an open-source tool provided by the Espressif IoT Development Framework (ESP-IDF) [14].

4.1 ESP32 Memory Structure

The flash storage of the ESP32 is responsible for storing multiple applications through various partition configurations. Specifically, Espressif provides two built-in partition tables: (i) 'Single factory app, no OTA' that can only store factory apps, and (ii) 'Factory app, two OTA definitions', designed to support OTA updates [15]. The specific ESP32 being examined was configured with the latter partition table, enabling OTA functionality. The sturcture of this partition table is illustrated in Figure 7.

Following the execution of an OTA update, the updated data is written to one of the app partitions that is currently

inactive. The partition ID associated with the updated data is stored in the otadata area. Upon successful completion of the update process, the ESP32 is undergoes an automatic reboot. During this reboot, the bootloader refers to the ID stored in the otadata to execute the corresponding partition [16]. In the event that the OTA data area is empty, the bootloader executes the factory partition, which contains the default application data.



(Figure 7) Partition table of ESP32

4.2 Memory Dump Analysis

In order to extract the contents of dump the flash storage, a connection is established between the ESP32 chip and a PC using UART communication. The ESP32 is equipped with a UART port which enables serial communication for tasks such as firmware uploading or monitoring output [17]. To perform the flash dump, the esptool.py is utilized. This tool allows for the extraction of the flash storage as a binary file. To identify traces of the attack, a total of three memory dumps were performed. Two dumps were executed during normal update attempts, each of which was given the name "*esp32_dump.bin*" and "*esp32_dump2.bin*" respectively. Additionally, a memory dump was taken after the SYN flooding attack and labeled as "*synflooding.bin*".

4.2.1 Analyzing *esp32_dump.bin*

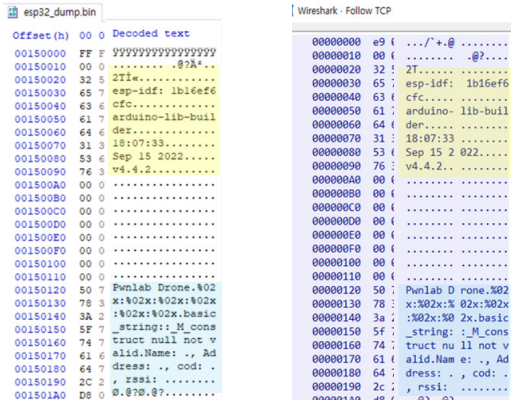
As part of the analysis to identify the memory relevant to the OTA update, the otadata partition within the *esp32_dump.bin file* was examined. It was discovered that this area contains a historical record of WiFi connections, which includes a collection of SSIDs (Service Set Identifiers) and their corresponding passwords. This information is depicted in Figure 8.

(Figure 8) WiFi data is stored in otadata partition (offset 0xD000)

Furthermore, it was revealed that the updated firmware content was written starting from the offset 0x150000 instead of the initial offset of app_0, which is 0x110000. To investigate this further, the updated firmware content compared with the packets sniffed from the perspective of an attacker above. Consequently, it was observed that the same values as the sniffed packet data were stored in the memory locations ranging from 0x150010 to 0x243D89, as shown in Figure 9.

This analysis implies that memory forensics enables the retrieval of a complete history of WiFi connections from the ESP32 drone. Not only does this apply to the attack case discussed in this paper, but this information can also serve as a valuable trace for identifying an attacker in various attack

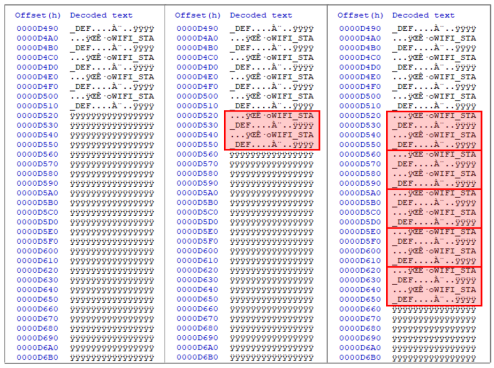
scenarios. Additionally, this information can provide crucial clues for detecting potential artifacts associated with the transmission of fake firmware.



(Figure 9) Comparing the esp32_dump.bin file (left) with the sniffed packet data (right)

4.2.2 Detecting evidence of TCP SYN Flooding

In order to identify memory artifacts of the SYN flooding attack, a comparison was made between the three dump files using WinMerge [18], a open-source file comparison tool. Figure 10 illustrates the comparision of the content within the files: *esp32_dump.bin*, *esp32_dump1.bin*, and *synflooding.bin*.



(Figure 10) The comparison was conducted between the following files: *esp32_dump.bin*, *esp32_dump1.bin*, and *synflooding.bin* (in the given order).

The analysis showed that the otadata partition recorded data block size of 0x40 during a regular OTA update. On the other hand, in the case of a TCP SYN flooding attack executed on the device, the memory registered five times the amount of data compared to a normal OTA update. This result suggests that an abnormally high number of the data blocks written in this partition can be considered an artifact of a SYN flooding attack.

Moreover, within the recorded data block, the field labeled “WIFI_STA” indicates the operational mode of the ESP32 device, functioning as a station and establishing an connection with an access point [19]. Considering that the block includes the device’s access information to an access point (WIFI_STA_DEF), it becomes plausible to regard this as a potential indication of other types of Denial-of-Service (DoS) attacks that can disrupt network connectivity.

5. Summary

IoT is playing an integral role in the hyper-connected era, and at the same time, various attack vectors such as malware threats and network vulnerabilities are on the rise. This study focuses on the ESP32 SoC, widely utilized in various IoT devices, and implements an attack scenario that disrupts the latest firmware updates through a DoS attack on OTA. Additionally, the forensic analysis of the ESP32 drones was conducted from a memory forensics perspective after the attacks, revealing traces of the DoS attack and demonstrating potential forensic evidence.

According to [20], unpatched vulnerabilities represent the primary attack vector and more than 50% of the 233 older vulnerabilites before 2021 have been exploited by ransomware groups. The simulated OTA attack on the IoT drone highlights the criticality of OTA availability by revealing the potential consequences of leaving vulnerabilities unpatched. This scenario exposes the drone to a range of potential threats, emphasizing the significance of maintaining a secure and up-to-date OTA system to safeguard against future risks.

References

[1] The Korea Herald, “IoT devices hacking statistics”, http://news.koreaherald.com/view.php?ud=20220119000736

[2] Yonhap News, “Home cameras hacking statistics”, https://en.yna.co.kr/view/AEN20221220009100315

[3] H. Jeon, and S. Lee, ″Analysis of Remote Update Vulnerabilities of IoT Healthcare Devices,″ Journal of KIIT, Vol. 19, No. 1, pp. 87-97, 2021. http://dx.doi.org/10.14801/jkiit.2021.19.1.87

[4] Espressif, https://www.espressif.com/

[5] Espressif, “ESP32-S2 Series Datasheet”, https://www.espressif.com/en/products/devkits

[6] O. Barybin, E. Zaitseva, and V. Brazhnyi, ″Testing the Security ESP32 Internet of Things Devices″, 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 2019. http://dx.doi.org/10.1109/PICST47496.2019.9061269

[7] Z. Li, H. Ren, E. Chou, X. Liu, and C. D. McAllister, ″Retrieving Forensically Sound Evidence from the ESP Series of IoT Devices″, IEEE Internet of Things Journal, Vol. 9, No.15, pp. 13144-13152, 2022. http://dx.doi.org/10.1109/JIOT.2022.3144164

[8] Aircrack-Ng,https://www.kali.org/tools/aircrack-ng/

[9] Wordlists,https://www.kali.org/tools/wordlists/

[10] Ettercap Project, https://www.ettercap-project.org/

[11] Wireshark, https://www.wireshark.org

[12] Hping3, https://www.kali.org/tools/hping3/

[13] Espressif, “ESP32-WROOM-32 Datasheet”, https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[14] Espressif, “Esptool.py Documentation”, https://docs.espressif.com/projects/esptool/en/latest/esp32/

[15] Espressif, “ESP-IDF API Guides”, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/partition-tables.html

[16] ESP32 Tutorials, http://www.lucadentella.it/en/2016/12/22/esp32-4-flash-bootloader-e-freertos/

[17] B. Pearson, L. Luo, Y. Zhang, R. Dey, Z. Ling, M. Bassiouni, and X. Fu, ″On Misconception of Hardware and Cost in IoT Security and Privacy″, In ICC 2019-2019 IEEE International Conference on Communications (ICC), 2019. http://dx.doi.org/10.1109/ICC.2019.8761062

[18] Winmerge, https://winmerge.org/

[19] ESP32 Networking APIs. Espressif, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html

[20] Hackers are increasingly targeting Zero-Day Vulnerabilites. Business Wire, https://www.businesswire.com/news/home/20220126005014/en/Ransomware-2021-Year-End-Report-Reveals-Hackers-are-Increasingly-Targeting-Zero-Day-Vulnerabilities-and-Supply-Chain-Networks-for-Maximum-Impact

◐ 저 자 소 개 ◑

백 지 연(Ji-yeon Baek)
2020년 성신여자대학교 융합보안공학과(공학사)
2023년 성신여자대학교 대학원 미래융합기술공학과(공학석사)
관심분야 : Automotive security, IoT security, etc.
E-mail : hesedbaek98@gmail.com

장 지 원(Jiwon Jang)
2022년 성신여자대학교 융합보안공학과(공학사)
2022년 성신여자대학교 심리학과(문학사)
2024년 성신여자대학교 대학원 미래융합기술공학과(공학석사)
관심분야 : Information Security, Pentest, IoT Security, etc.
E-mail : nebulaboratory@gmail.com

김 성 민(Seongmin Kim)
2012년 한국과학기술원 전기 및 전자공학과 졸업
2014년 한국과학기술원 전기 및 전자공학과 석사
2019년 한국과학기술원 정보보호대학원 박사
2020년 9월~현재: 성신여자대학교 융합보안공학과 조교수
관심분야 : 신뢰 실행 환경, 클라우드 컴퓨팅, 시스템 보안
E-mail : sm.kim@sungshin.ac.kr