International Journal of Internet, Broadcasting and Communication Vol.9 No.3 9-16 (2017)
https://doi.org/10.7236/IJIBC.2017.9.3.9

IJIBC 17-3-2

Study on Structural and Systematic Security Threats of Vehicle Black Box as Embedded System

Jaehyun Park[1], WoongChul Choi[2*]

[1,2]*Dept. of Computer Science, KwangWoon University*
[1]*bobjaehyunpark@gmail.com*, [*]*wchoi@kw.ac.kr*

Abstract

Recently, more users have been using IoT embedded systems. Since the wireless network function is a basic and core function in most embedded systems, new security threats and weaknesses are expected to occur. In order to resolve these threats, it is necessary to investigate the security issues in the development stages according to the Security Development Lifecycle (SDL). This study analyzes the vulnerabilities of the embedded systems equipped with the wireless network function, and derives possible security threats and how dangerous such threats are. We present security risks including bypassing the authentication stage required for accessing to the embedded system.

Keywords: *Embedded System Security, Security Threat, Security Vulnerabilities*

1. Introduction

IoT embedded systems can hold many kinds of information inside, from personal information to privacy and sensitive information. Personal conversations and important decision-making procedures can also be the sensitive information. In the IoT systems, the wireless networking function is generally a core function, therefore, security threats and vulnerabilities can occur in such IoT systems. This study shows possible security risks with exploiting security vulnerabilities and structural problems of embedded systems. In order to prevent and resolve such problems, we propose security measures against the problems of the embedded systems.

2. Security Vulnerabilities and Risk

2.1 Environment Settings for Analyzation

Before security vulnerabilities and threats of the embedded system are analyzed, an environment for the analysis needs to be set up in the system. First, we need a virtual environment called QEMU. QEMU is a virtual environment for analyzing dynamically the firmware of the embedded system. The firmware can be extracted in several ways. In this study, we extract the firmware using a UART port. Most UART ports are used for debugging. One UART port is connected to a serial communication link to obtain firmware. Such an

Manuscript Received: May. 16, 2017 / Revised: May. 27, 2017 / Accepted: June. 6, 2017
*Corresponding Author: wchoi@kw.ac.kr
Tel: +82-2-940-5582, Fax: +82-2-940-5582
Dept. of Computer Science, KwangWoon University, Korea

experimental setup is normally used for analyzing the vulnerabilities of embedded systems which have ARM or MIPS processors, and it is useful for static and dynamically analyze through various functions of environment.

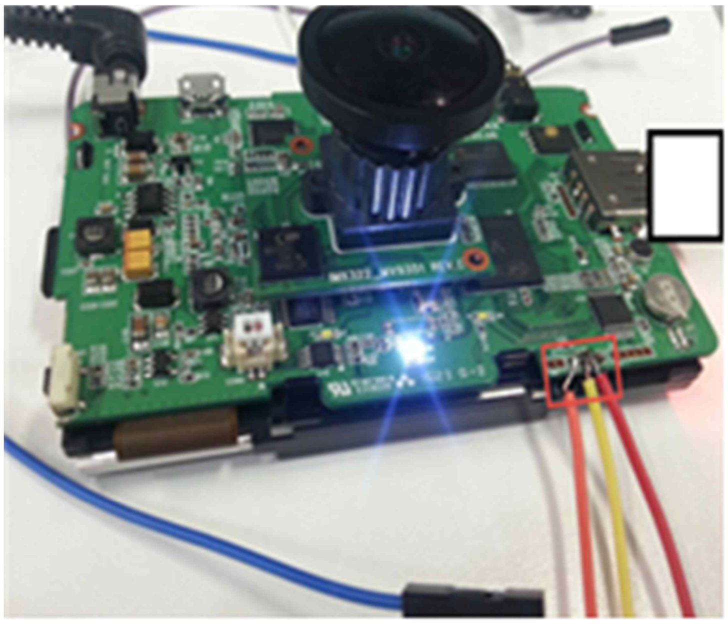
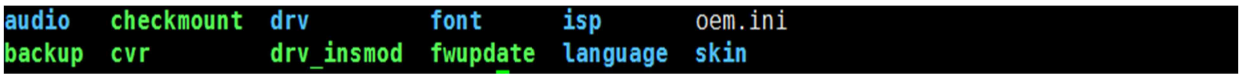


Figure 1. Connecting with UART Port

Most embedded systems can also use several methods for firmware extraction. Downloading the firmware from the official website and extracting it using extraction tools such as FMK is one of the typical examples. In addition, we can use the UART port to extract the firmware through the serial communication link as shown in Fig. 1, and can utilize the JTAG and flash memory dump. Once we extract the firmware, we can see the root shell shown as in the Figure 2.



```
audio   checkmount  drv        font      isp       oem.ini
backup  cvr         drv_insmod fwupdate  language  skin
```

Figure 2. Extracting Root Shell

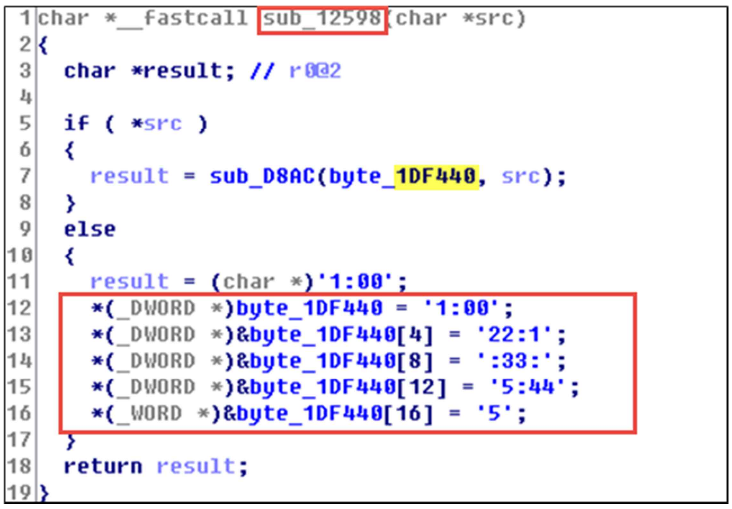
2.2 Structural & Systematic Vulnerabilities

In this study, we classify the vulnerabilities into two broad categories - structural vulnerability and systematical vulnerability. The systematic vulnerability refers to the vulnerability such as buffer over flow caused by memory corruption, and the structural vulnerability means the vulnerability caused by miss-designed in the level of development of the software, i.e., during software development lifecycle, security issues are not considered, which could lead the security vulnerabilities.

Many IoT systems and devices, including the systems with wireless communication functions, are basically equipped with features that allow them to connect to personal smartphones. Using this feature, users need to be authenticated to access into the system. There are a variety of methods that users can use for authentication. One simple way to do so is to register the device in advance in the IoT system with MAC address of the device. While registering MAC address of a device in the IoT system is an easy way for authentication, possible security issues can occur, which are presented in the following.

2.3 Examples of Structural Vulnerabilities

Authentication using MAC addresses can cause many problems and security threats. Normally, before releasing IoT products, there is a possibility that the default MAC address remains in the product by accident. Figure 3 shows that the default MAC address remains in the firmware of the embedded system, and all the functions implemented in the embedded system can be executed using this default MAC address with proper authentication procedure. Also, even if such a default MAC address does not exist, there is still a possibility that the system is exposed to the packet sniffing security risk. Normally, using a packet sniffing tool, all the communicating packets with a client user can be sniffed. But using the special mode, called the monitor mode of the device, all the communicating packets, not only with the client user and the embedded system, but also the client's communicating packet with other devices or even with users can be sniffed. When other users try to have authentication progress, attackers are able to sniff the authenticating packet with utilizing monitor mode, and also sniff the user's MAC address which is used in authentication. With using MAC address which is sniffed with utilizing monitor mode sniffer, attacker can not only access to the system, but can also use all the implemented functions.



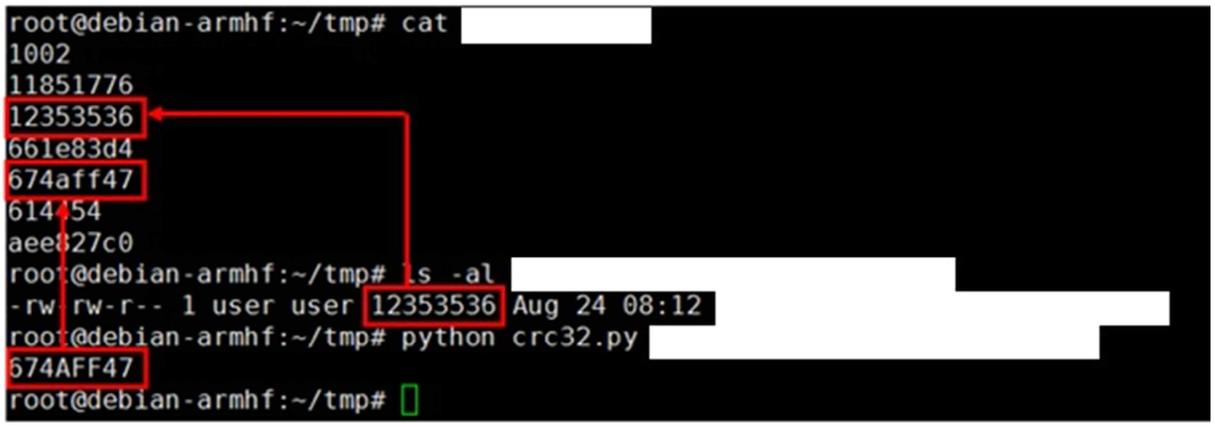
```
char *__fastcall sub_12598(char *src)
{
  char *result; // r0@2

  if ( *src )
  {
    result = sub_D8AC(byte_1DF440, src);
  }
  else
  {
    result = (char *)'1:00';
    *(_DWORD *)byte_1DF440 = '1:00';
    *(_DWORD *)&byte_1DF440[4] = '22:1';
    *(_DWORD *)&byte_1DF440[8] = ':33:';
    *(_DWORD *)&byte_1DF440[12] = '5:44';
    *(_WORD *)&byte_1DF440[16] = '5';
  }
  return result;
}
```

Figure 3. Default MAC Address Vulnerability

Embedded systems have security threats in terms of firmware. Basically, embedded systems are able to update the firmware remotely. The update can be performed using a smart phone, also user's PC, and SD cards. Firmware updates are also important part of embedded system's security. In the event of a security vulnerability, it is very important to envision patches quickly and require all users to update their firmware. However, security threats can be also derived in how the firmware updates are performed.

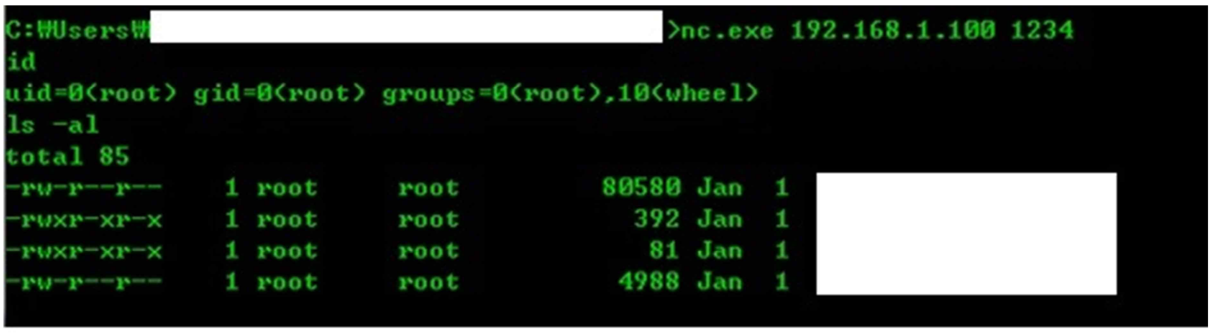
The embedded system used in this study is able to update the firmware remotely. The problem is in the process of checking the integrity of the firmware, which relies on the integrity of the certain values of the binary(firmware). The embedded system used in this study depends on the CRC32 value of the firmware and the file size. In this case, malicious firmware can be updated on the device if malicious firmware is changed to match only those values. In this case, malicious firmware means the firmware that backdoor is installed.

Being able to update maliciously modified firmware, including firmware with installed backdoors, is dangerous from a security standpoint because it means that the system kernel could be intentionally changed. Following Figure 4 shows the parts checked on the embedded device when the firmware is updated. And Figure 8 shows the backdoor installed on the embedded device by updating the firmware with the backdoor installed.



```
root@debian-armhf:~/tmp# cat [redacted]
1002
11851776
12353536
661e83d4
674aff47
614454
aee827c0
root@debian-armhf:~/tmp# ls -al [redacted]
-rw-rw-r-- 1 user user 12353536 Aug 24 08:12 [redacted]
root@debian-armhf:~/tmp# python crc32.py [redacted]
674AFF47
root@debian-armhf:~/tmp# 
```

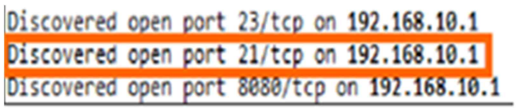
Figure 4. Firmware Update Vulnerability



```
C:₩Users₩[redacted]>nc.exe 192.168.1.100 1234
id
uid=0(root) gid=0(root) groups=0(root),10(wheel)
ls -al
total 85
-rw-r--r--    1 root     root        80580 Jan  1 [redacted]
-rwxr-xr-x    1 root     root          392 Jan  1
-rwxr-xr-x    1 root     root           81 Jan  1
-rw-r--r--    1 root     root         4988 Jan  1
```

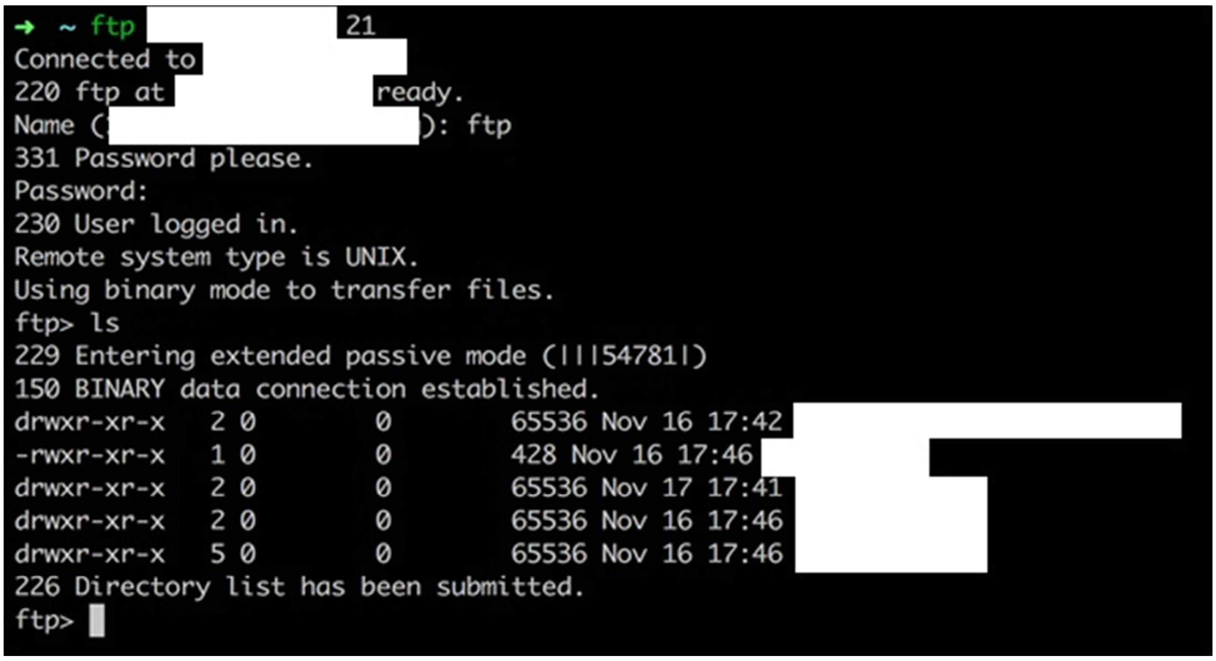
Figure 5. Backdoor with Malicious Firmware Update

In addition, embedded devices equipped with wireless network functions communicate with users in real time, so that ports that perform various roles are opened. This can also be a problem in the point of security. The port is open for services that are not used often, which can be a security problem. We can see which ports are being used(opened) through the port scanning, and the ports for unused services may be opened. In particular, TELNET port which has port number 23, and FTP port which has number 21 are dangerous in the point of security.



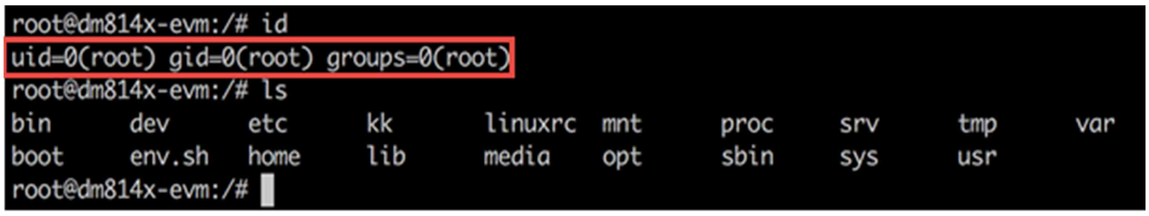
```
Discovered open port 23/tcp on 192.168.10.1
Discovered open port 21/tcp on 192.168.10.1
Discovered open port 8080/tcp on 192.168.10.1
```

Figure 6. Scanning FTP Port



```
➜ ~ ftp [redacted] 21
Connected to [redacted]
220 ftp at [redacted] ready.
Name ([redacted]): ftp
331 Password please.
Password:
230 User logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering extended passive mode (|||54781|)
150 BINARY data connection established.
drwxr-xr-x   2 0        0        65536 Nov 16 17:42 [redacted]
-rwxr-xr-x   1 0        0        428 Nov 16 17:46 [redacted]
drwxr-xr-x   2 0        0        65536 Nov 17 17:41 [redacted]
drwxr-xr-x   2 0        0        65536 Nov 16 17:46 [redacted]
drwxr-xr-x   5 0        0        65536 Nov 16 17:46 [redacted]
226 Directory list has been submitted.
ftp> █
```

Figure 7. Connecting with FTP Port



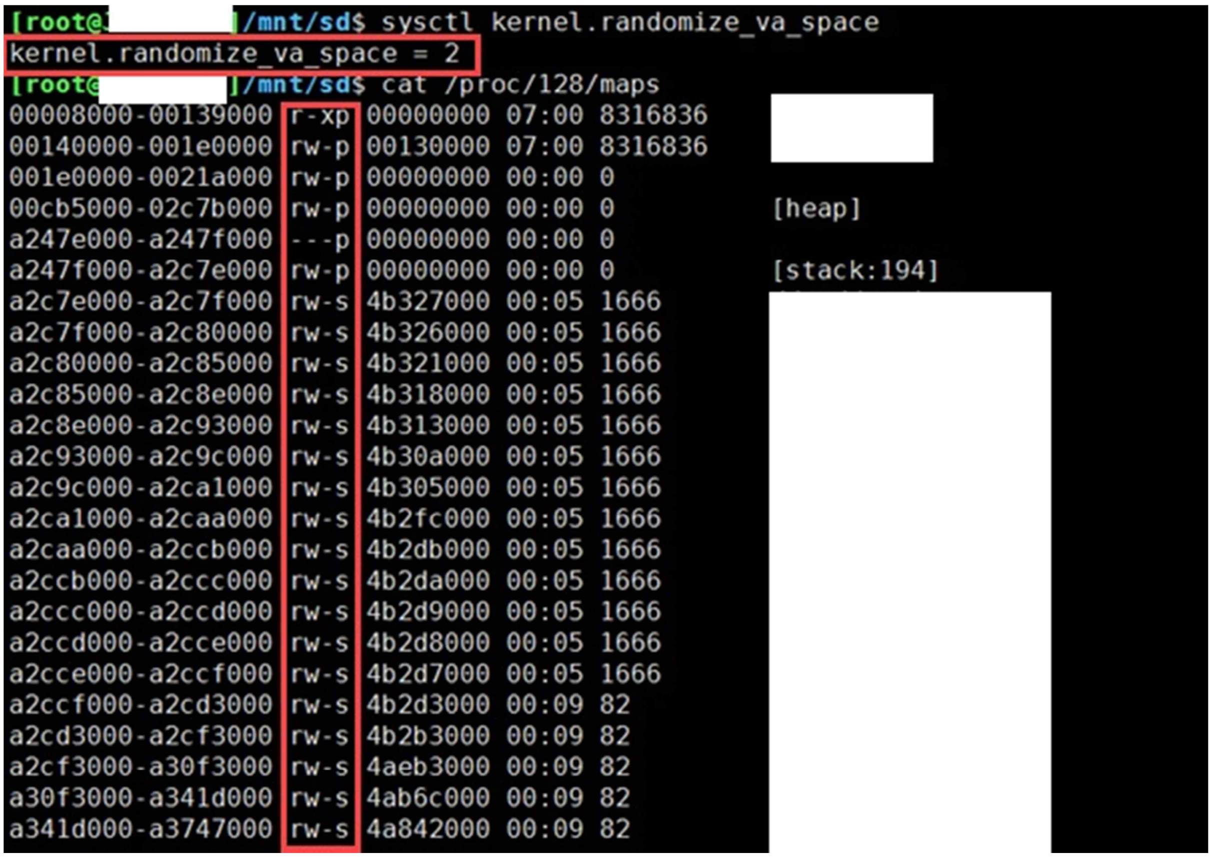
```
root@dm814x-evm:/# id
uid=0(root) gid=0(root) groups=0(root)
root@dm814x-evm:/# ls
bin      dev      etc      kk       linuxrc  mnt      proc     srv      tmp      var
boot     env.sh   home     lib      media    opt      sbin     sys      usr
root@dm814x-evm:/# █
```

Figure 8. Connect to Root Shell Using TELNET

2.4 Example of Systematic Vulnerabilities

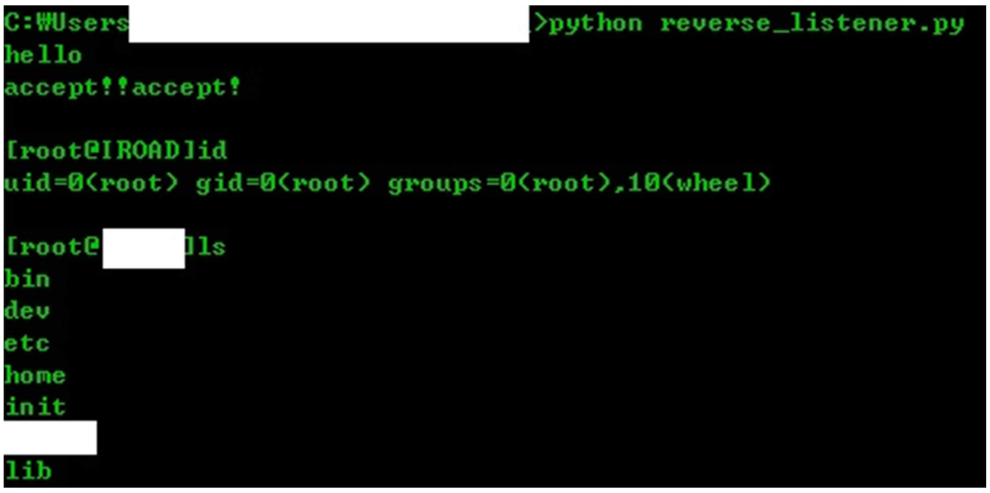
In addition to these structural vulnerabilities, systematic vulnerabilities due to memory corruption can also be very dangerous. Memory corruption has various vulnerabilities such as buffer over flow(BOF), use after free(UAF), and format string bug(FSB), and more. The vulnerability in this study is a buffer over flow vulnerability. Various functions are implemented in the embedded device. In order to use the functions, various input values are received from the user. At this time, the input value may cause an error in the apparatus, systematically invading another range, and executing malicious code. We call this input as “Untrusted Input”.

Basically, various systems including embedded system have various security techniques to prevent the memory corruption(access violation). Typically, there are ASLR(Address Space Layout Randomization), DEP(Data Execution Prevention)/NX(Non-Execute) Bit, and ASCII ARMOR. However, in the case of DEP, it is possible to bypass with using RTL(Return To Library) technique. Also, in case of ASLR and ASCII ARMOR, it is possible to bypass with using ROP(Return Oriented Programming) attack technique. The important point here is that embedded devices with wireless networking capabilities can be remotely attacked because they can receive code remotely with using wireless network. This can lead to elevation of privilege by remote code execution and also can lead to a very dangerous security threats. Following Figure 9 shows the memory corruption prevention technique, which is ASLR and DEP. Word “kernel randomize” shows this system have ASLR protection. Also, there is no character “x” which means the execution privilege, we can know that this system doesn’t give execution privilege in stack and heap area.



```
[root@[redacted]]/mnt/sd$ sysctl kernel.randomize_va_space
kernel.randomize_va_space = 2
[root@[redacted]]/mnt/sd$ cat /proc/128/maps
00008000-00139000 r-xp 00000000 07:00 8316836     [redacted]
00140000-001e0000 rw-p 00130000 07:00 8316836
001e0000-0021a000 rw-p 00000000 00:00 0
00cb5000-02c7b000 rw-p 00000000 00:00 0          [heap]
a247e000-a247f000 ---p 00000000 00:00 0
a247f000-a2c7e000 rw-p 00000000 00:00 0          [stack:194]
a2c7e000-a2c7f000 rw-s 4b327000 00:05 1666       [redacted]
a2c7f000-a2c80000 rw-s 4b326000 00:05 1666
a2c80000-a2c85000 rw-s 4b321000 00:05 1666
a2c85000-a2c8e000 rw-s 4b318000 00:05 1666
a2c8e000-a2c93000 rw-s 4b313000 00:05 1666
a2c93000-a2c9c000 rw-s 4b30a000 00:05 1666
a2c9c000-a2ca1000 rw-s 4b305000 00:05 1666
a2ca1000-a2caa000 rw-s 4b2fc000 00:05 1666
a2caa000-a2ccb000 rw-s 4b2db000 00:05 1666
a2ccb000-a2ccc000 rw-s 4b2da000 00:05 1666
a2ccc000-a2ccd000 rw-s 4b2d9000 00:05 1666
a2ccd000-a2cce000 rw-s 4b2d8000 00:05 1666
a2cce000-a2ccf000 rw-s 4b2d7000 00:05 1666
a2ccf000-a2cd3000 rw-s 4b2d3000 00:09 82
a2cd3000-a2cf3000 rw-s 4b2b3000 00:09 82
a2cf3000-a30f3000 rw-s 4aeb3000 00:09 82
a30f3000-a341d000 rw-s 4ab6c000 00:09 82
a341d000-a3747000 rw-s 4a842000 00:09 82
```

Figure 9. ASLR and DEP Protection



```
C:₩Users[redacted]>python reverse_listener.py
hello
accept!!accept!

[root@IROAD]id
uid=0(root) gid=0(root) groups=0(root),10(wheel)

[root@[redacted]]ls
bin
dev
etc
home
init
[redacted]
lib
```

Figure 10. Threat of Buffer Over Flow Vulnerability

3. Conclusion

In this study, we divide vulnerabilities of embedded systems into two categories of structural and systematic security vulnerabilities. It can be shown that the features of these two vulnerabilities are quite different. In addition, there is much difference in the prevention methods for these two types of vulnerabilities. Firstly, in the case of structural vulnerabilities, it is important to detect and prevent security threats in the stage of designing a system. SDL(Secure Development Lifecycle) can be utilized. SDL is a software development process used and proposed by Microsoft. Secondly, in the case of systematic vulnerabilities, it is important to apply security patches through diagnosis of steady vulnerabilities. While it is important to consider the security aspects such as utilizing the SDL(Secure Development Lifecycle) in the designing level, the security factors for embedded systems which are already released for wireless network functions should be considered as well.

Embedded devices are not simple devices with wireless networking capabilities. Rather, it holds important information, including various sensitive personal information. As security issues such as privacy and personal information are increasing, the security aspects of embedded systems with wireless networking capabilities should be researched steadily.

Acknowledgements

The present Research has been conducted by the Research Grant of Kwangwoon University in 2015.

4. Reference

[1] Brendan O'Connor, "Vulnerabilities in Not-So Embedded Systems", Black Hat USA, 2006

[2] P. Marwedel, "Embedded system design: Embedded systems foundations of cyber-physical systems", Springer Science, 2010

[3] B. Schneier, "Security risks of embedded systems", 2014

[4] D. Kleidermacher, M Kleidermacher, "Embedded systems security: practical methods for safe and secure software and systems development", Elsevier, 2012

[5] P. Koopman, "Embedded system security", Computer, vol. 37 no. 7, pp.95-97, 2004

[6] Jin-bing Hou, Tong Li, Cheng Chang, "Research for Vulnerability Detection of Embedded System Firmware", 2017

[7] Sri Parameswaran, Tilman Wolf, “Embedded systems security-an overview”, 2008

[8] Michael horkan, “A Black-Box Approach to Embedded Systems Vulnerability Assessment”, 2016

[9] Youngsuk Kim, Youngmin Kwon, “Implementation of Embedded POC UA Server for interoperation in EtherCAT”, 2016

[10] Dorottya Papp , Zhendong Ma, Levente Buttyan, "Embedded systems security : Threats, vulnerabilities, and attack taxonomy".

[11] Somesh Jha, "Software Security Issuses in Embedded Systems".

[12] Vijaykrishnan Narayanan, Yuan Xie, “Reliability Concerns in Embedded System Designs”, 2006

[13] Vijeet H. Meshram, Ashish B. Sasankar, “Security in Embedded Systems: Vulnerabilities, Pigeonholing of Attacks and Countermeasures”, 2011