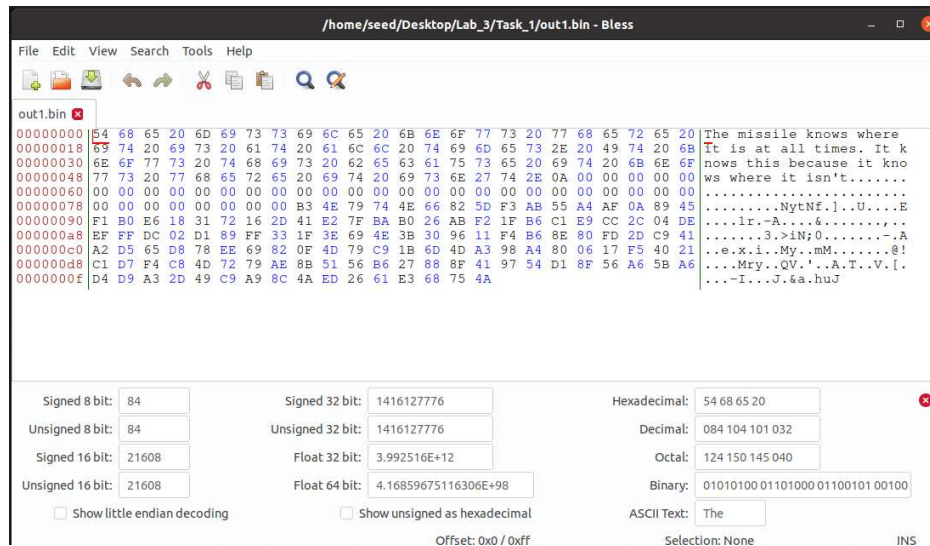


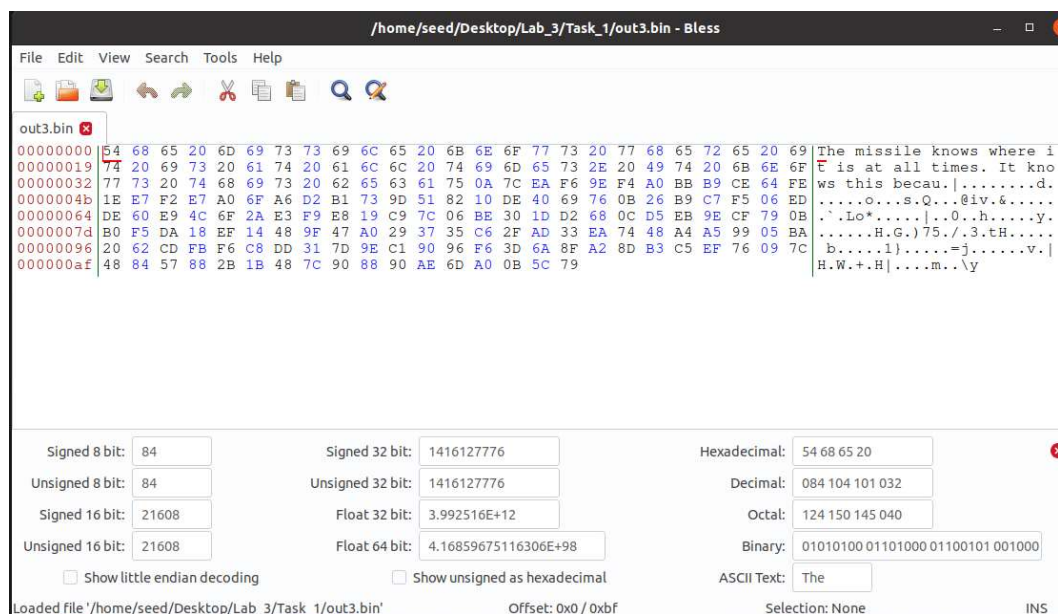
Lab 3

Task 1

- If the length of prefix file is not multiple of 64 bytes, there will be '0' padding for the binary file.



- If the prefix file is exactly 64 bytes, there will be no padding for the binary file.



- Data generated by md5collgen only have slightly differences. Byte 60, 112, and 125 have difference.

The screenshot shows the Bless tool interface with the following data:

File	Address	Hex	Dec	Bin	ASCII
out4.bin	00000000	54 68 65 20 6D 69 73 73 69 6C 65 20 6B 6E 6F 77 73 20 77 68	86 101 101 32 109 108 80 110 111 101 101 101 101 101 101 101 101	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	The missile knows wh
out3.bin	77 73 20 77 68	6C 20 74 69 6D 69 73 20 62 65 69 73 20 62 65 69 73 20 62 65	108 101 32 109 108 80 110 111 101 101 101 101 101 101 101 101	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	The missile knows wh

Conversion options at the bottom:

- Signed 8 bit: 11
- Unsigned 8 bit: 11
- Signed 16 bit: 2992
- Unsigned 16 bit: 2992
- Signed 32 bit: 196146650
- Unsigned 32 bit: 196146650
- Float 32 bit: 6.816265E-32
- Float 64 bit: 2.31333584297936E-252
- Hexadecimal: 0B B0 F5 DA
- Decimal: 011 176 245 218
- Octal: 013 260 365 332
- Binary: 00001011 10110000 11110110
- ASCII Text: [Empty]
- Selection: 0x7b to 0x7b (0x1 bytes)

The screenshot shows the Bless tool interface with the following data:

File	Address	Hex	Dec	Bin	ASCII
out4.bin	00000000	54 68 65 20 6D 69 73 73 69 6C 65 20 6B 6E 6F 77 73 20 77 68	86 101 101 32 109 108 80 110 111 101 101 101 101 101 101 101 101	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	The missile knows wh
out3.bin	77 73 20 77 68	6C 20 74 69 6D 69 73 20 62 65 69 73 20 62 65 69 73 20 62 65	108 101 32 109 108 80 110 111 101 101 101 101 101 101 101 101	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	The missile knows wh

Conversion options at the bottom:

- Signed 8 bit: -96
- Unsigned 8 bit: 160
- Signed 16 bit: -24565
- Unsigned 16 bit: 40971
- Signed 32 bit: -1609868167
- Unsigned 32 bit: 2685099129
- Float 32 bit: -1.180435E-19
- Float 64 bit: --
- Hexadecimal: A0 0B 5C 79
- Decimal: 160 011 092 121
- Octal: 240 013 134 171
- Binary: 10100000 00001011 01011110
- ASCII Text: [Empty]
- Selection: 0xbb to 0xbb (0x1 bytes)

The screenshot shows the Bless tool interface with the following data:

File	Address	Hex	Dec	Bin	ASCII
out4.bin	00000000	54 68 65 20 6D 69 73 73 69 6C 65 20 6B 6E 6F 77 73 20 77 68	86 101 101 32 109 108 80 110 111 101 101 101 101 101 101 101 101	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	The missile knows wh
out3.bin	77 73 20 77 68	6C 20 74 69 6D 69 73 20 62 65 69 73 20 62 65 69 73 20 62 65	108 101 32 109 108 80 110 111 101 101 101 101 101 101 101 101	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	The missile knows wh

Conversion options at the bottom:

- Signed 8 bit: -96
- Unsigned 8 bit: 160
- Signed 16 bit: -24565
- Unsigned 16 bit: 40971
- Signed 32 bit: -1609868167
- Unsigned 32 bit: 2685099129
- Float 32 bit: -1.180435E-19
- Float 64 bit: --
- Hexadecimal: A0 0B 5C 79
- Decimal: 160 011 092 121
- Octal: 240 013 134 171
- Binary: 10100000 00001011 01011110
- ASCII Text: [Empty]
- Selection: 0xbb to 0xbb (0x1 bytes)

Task 2

First, I create 2 text files then md5collgen them. Run **md5sum** Then I append new text to those md5 binary files. Run **md5sum** to both files again. It appears those md5 hash remains same.

```
seed@VM: ~/Task_2
[02/23/23]seed@VM:~/Task_2$ md5collgen -p Missile.txt -o out1 out2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1' and 'out2'
Using prefixfile: 'Missile.txt'
Using initial value: cb3e0fe75d5a0b018f6ca56c4c94ba9f

Generating first block: .
Generating second block: S11.....
Running time: 0.612241 s
[02/23/23]seed@VM:~/Task_2$ md5sum out1 out2
de5b7cfa97713dec8edfac92b796f609 out1
de5b7cfa97713dec8edfac92b796f609 out2
[02/23/23]seed@VM:~/Task_2$ cat out1 Aim9L > out3
[02/23/23]seed@VM:~/Task_2$ cat out2 Aim9L > out4
[02/23/23]seed@VM:~/Task_2$ md5sum out3 out4
37197e65178e2c3ce61cea19b5dd5d20 out3
37197e65178e2c3ce61cea19b5dd5d20 out4
[02/23/23]seed@VM:~/Task_2$ md5sum out1 out2
de5b7cfa97713dec8edfac92b796f609 out1
de5b7cfa97713dec8edfac92b796f609 out2
[02/23/23]seed@VM:~/Task_2$ md5sum Missile.txt
a48cc9d88734ac99bb463dc8a92283e6 Missile.txt
[02/23/23]seed@VM:~/Task_2$
```

The top screenshot shows a hex editor window titled "/home/seed/Desktop/Lab_3/Task_2/out3 - Bless". It displays a hex dump of a file named "out3". The hex data is shown in columns, with corresponding ASCII text on the right. The ASCII text is a repeating pattern of "The missile knows where it is at all times. It knows this because it knows where it isn't." followed by various characters and symbols.

The bottom screenshot shows a conversion tool window titled "/home/seed/Desktop/Lab_3/Task_2/out4 - Bless". It displays a hex dump of a file named "out4". The hex data is shown in columns, with corresponding ASCII text on the right. The ASCII text is a repeating pattern of "The missile knows where it is at all times. It knows this because it knows where it isn't." followed by various characters and symbols.

Below the hex editor, there is a conversion tool window. It contains several input fields for different data types: Signed 8 bit, Unsigned 8 bit, Signed 16 bit, Unsigned 16 bit, Signed 32 bit, Unsigned 32 bit, Float 32 bit, Float 64 bit, Hexadecimal, Decimal, Octal, Binary, and ASCII Text. The tool also has checkboxes for "Show little endian decoding" and "Show unsigned as hexadecimal". The "Offset" is set to "0x0 / 0x106" and the "Selection" is set to "None".

Task 3

I used C program to create a 200 bytes string array. Then count 12352 bytes as multiple of 64. In addition, that truncates a part of array to the suffix. Then, cut 128 bytes after prefix as p and q; after 128 bytes, the rest of the file is suffix. Concatenate prefix, p/q(as 2 version of final output), and suffix.

```
seed@VM: ~/.../Task_3
[02/24/23]seed@VM:~/.../Task_3$ head -c 12352 MD5Missile.o > prefix
[02/24/23]seed@VM:~/.../Task_3$ md5collgen -p prefix -o md5prefix_1.bin md5prefix_2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'md5prefix_1.bin' and 'md5prefix_2.bin'
Using prefixfile: 'prefix'
Using initial value: 38601c58a31395fe26a1f08a528dc558

Generating first block: .....
Generating second block: S01.....
Running time: 6.00462 s
[02/24/23]seed@VM:~/.../Task_3$ tail -c +12480 MD5Missile.o > suffix
[02/24/23]seed@VM:~/.../Task_3$ tail -c 128 md5prefix_1.bin > p
[02/24/23]seed@VM:~/.../Task_3$ tail -c 128 md5prefix_2.bin > q
[02/24/23]seed@VM:~/.../Task_3$ cat prefix p suffix > final_1
[02/24/23]seed@VM:~/.../Task_3$ cat prefix q suffix > final_2
[02/24/23]seed@VM:~/.../Task_3$ diff final_1 final_2
bash: syntax error near unexpected token `)'
[02/24/23]seed@VM:~/.../Task_3$ diff final_1 final_2
Binary files final_1 and final_2 differ
[02/24/23]seed@VM:~/.../Task_3$ md5sum final_1 final_2
0b4373f317c828519289578b8ec3f3a6  final_1
0b4373f317c828519289578b8ec3f3a6  final_2
[02/24/23]seed@VM:~/.../Task_3$
```

After concatenation, both final version still shares same md5 hash despite the 128 bytes part is completely different from the original ones.

```
seed@VM: ~/.../Task_3
[02/24/23]seed@VM:~/.../Task_3$ head -c 12352 MD5Missile.o > prefix
[02/24/23]seed@VM:~/.../Task_3$ md5collgen -p prefix -o md5prefix_1.bin md5prefix_2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'md5prefix_1.bin' and 'md5prefix_2.bin'
Using prefixfile: 'prefix'
Using initial value: 38601c58a31395fe26a1f08a528dc558

Generating first block: .....
Generating second block: S01.....
Running time: 6.00462 s
[02/24/23]seed@VM:~/.../Task_3$ tail -c +12480 MD5Missile.o > suffix
[02/24/23]seed@VM:~/.../Task_3$ tail -c 128 md5prefix_1.bin > p
[02/24/23]seed@VM:~/.../Task_3$ tail -c 128 md5prefix_2.bin > q
[02/24/23]seed@VM:~/.../Task_3$ cat prefix p suffix > final_1
[02/24/23]seed@VM:~/.../Task_3$ cat prefix q suffix > final_2
[02/24/23]seed@VM:~/.../Task_3$ diff final_1 final_2
bash: syntax error near unexpected token `)'
[02/24/23]seed@VM:~/.../Task_3$ diff final_1 final_2
Binary files final_1 and final_2 differ
[02/24/23]seed@VM:~/.../Task_3$ md5sum final_1 final_2
0b4373f317c828519289578b8ec3f3a6  final_1
0b4373f317c828519289578b8ec3f3a6  final_2
[02/24/23]seed@VM:~/.../Task_3$
```

[illegible][illegible]

Task 4

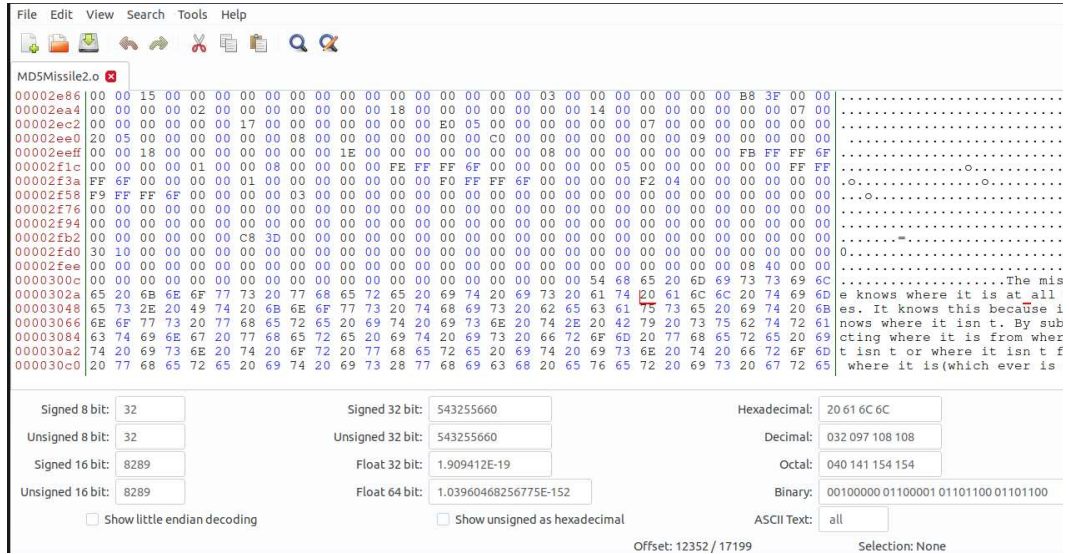
I create 2 arrays in C program. Just like previous one, count 12352 bytes then cut bytes before that byte as prefix. Then count 128 byte as p/q in X-array. After the 128 byte, starts the suffix part. However, 1st 96 bytes of the suffix is the 1st segment of suffix. Therefore, After 96 bytes of the suffix, the 128 bytes after that is Y-array. At this point, 128 + 96 bytes(1st segment of suffix + 128 byte Y-array part) leads to the starting point of rest of segment of suffix. Concatenate **prefix + p/q + 1st segment of suffix + p + rest of suffix** will lead to final_1 and final_2 program. Final_1 has **both p's**, and Final_2 has **p and q**. Run them, it will be shown that even though the p/q part has changed, if rest of the binary files are not tampered, the outcome is still same. If the array are different, then the terminal will shot 'Run Malicious Code' instead of 'run benign code'.

```
seed@VM: ~/Task_4
[02/24/23]seed@VM:~/Task_4$ head -c 12352 MD5Missile2.o > prefix
[02/24/23]seed@VM:~/Task_4$ md5collgen -p prefix -o md5out1.bin md5out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

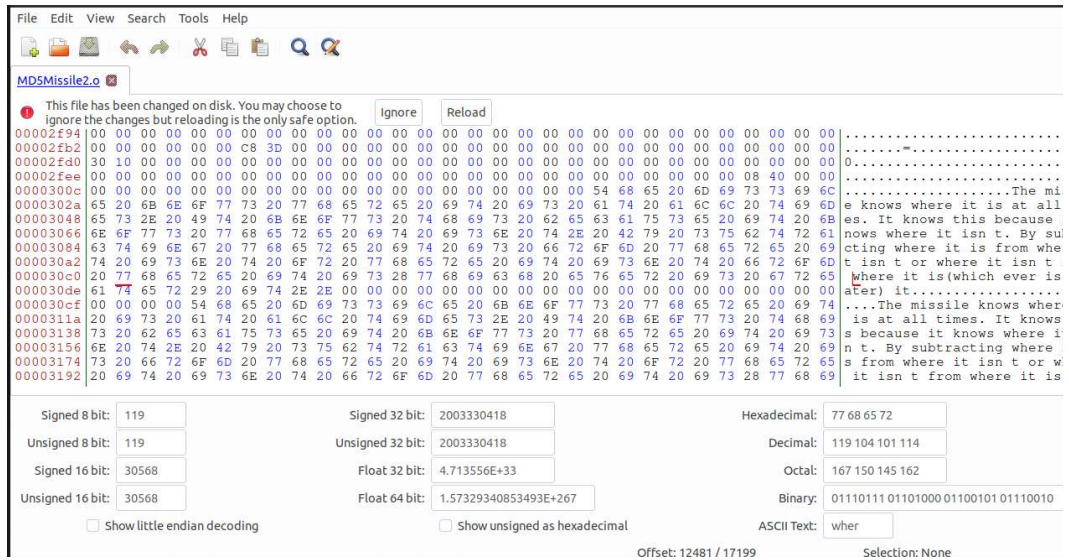
Using output filenames: 'md5out1.bin' and 'md5out2.bin'
Using prefixfile: 'prefix'
Using initial value: c9ef056824226e3de3eb7250ad1264a2

Generating first block: .....
Generating second block: 500.....
Running time: 18.5715 s
[02/24/23]seed@VM:~/Task_4$ tail -c 128 md5out1.bin > p
[02/24/23]seed@VM:~/Task_4$ tail -c 128 md5out1.bin > q
[02/24/23]seed@VM:~/Task_4$ tail -c 12481 MD5Missile2.o > suffix
[02/24/23]seed@VM:~/Task_4$ head -c 96 suffix > 1st_segment_suffix
[02/24/23]seed@VM:~/Task_4$ tail -c +12481 MD5Missile2.o > suffix
[02/24/23]seed@VM:~/Task_4$ head -c 96 suffix > 1st_segment_suffix
[02/24/23]seed@VM:~/Task_4$ tail -c +225 suffix > 2nd_segment_suffix
[02/24/23]seed@VM:~/Task_4$ cat prefix p 1st_segment_suffix p 2nd_segment_suffix > final_1
[02/24/23]seed@VM:~/Task_4$ cat prefix q 1st_segment_suffix p 2nd_segment_suffix > final_2
[02/24/23]seed@VM:~/Task_4$ chmod +x final_1
[02/24/23]seed@VM:~/Task_4$ ./final_1
Run benign code
[02/24/23]seed@VM:~/Task_4$ ./final_2
Run benign code
[02/24/23]seed@VM:~/Task_4$
```

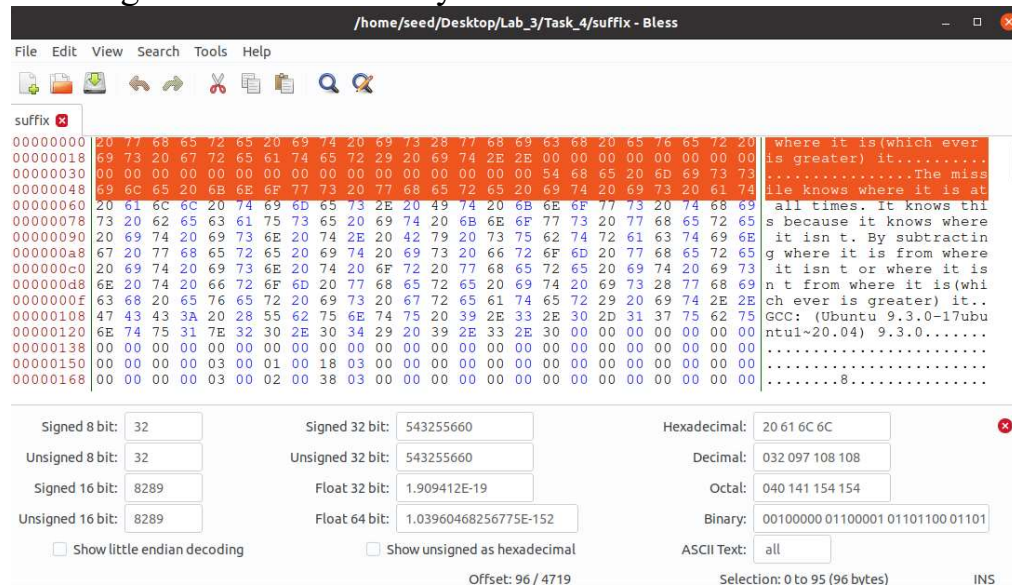

End of prefix, start of 1st p/q 128 byte part. 12352 byte point.



End of 1st p/q 128 byte part. Start of suffix. 12481 byte point.



First segment of suffix. 96 bytes.



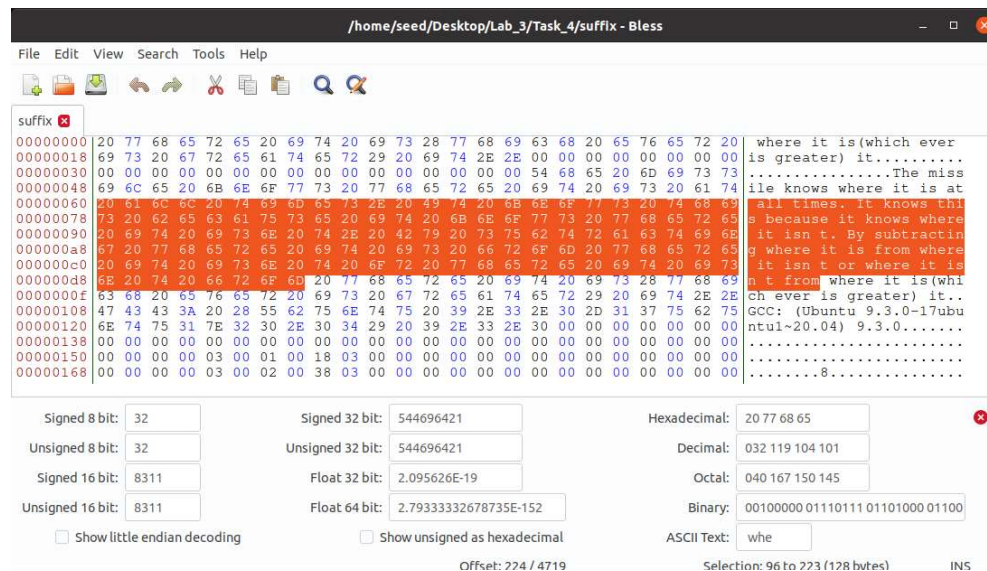
File Edit View Search Tools Help

suffix

00000000	20	77	68	65	72	65	20	69	74	20	69	73	28	77	68	69	63	68	20	65	76	65	72	20	where it is(which ever
00000018	69	73	20	67	72	65	61	74	65	72	29	20	69	74	2E	2E	00	00	00	00	00	00	00	00	is greater) it.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	54	68	65	20	6D	69	73	73The miss
00000048	69	6C	65	20	6B	6E	6F	77	73	20	77	68	65	72	65	20	69	74	20	69	73	20	61	74	ile knows where it is at
00000060	20	61	6C	6C	20	74	69	6D	65	73	2E	20	49	74	20	6B	6E	6F	77	73	20	74	68	69	all times. it knows thi
00000078	73	20	62	65	63	61	75	73	65	20	69	74	20	6B	6E	6F	77	73	20	77	68	65	72	65	s because it knows where
00000090	20	69	74	20	69	73	6E	20	74	2E	20	42	79	20	73	75	62	74	72	61	63	74	69	6E	it isn t. By subtractin
000000a8	67	20	77	68	65	72	65	20	69	74	20	69	73	20	66	72	6F	6D	20	77	68	65	72	65	g where it is from where
000000c0	20	69	74	20	69	73	6E	20	74	20	6F	72	20	77	68	65	72	65	20	69	74	20	69	73	it isn t or where it is
000000d8	6E	20	74	20	66	72	6F	6D	20	77	68	65	72	65	20	69	74	20	69	73	28	77	68	69	n t from where it is(whi
000000f0	63	68	20	65	76	65	72	20	69	73	20	67	72	65	61	74	65	72	29	20	69	74	2E	2E	ch ever is greater) it..
00000108	47	43	43	3A	20	28	55	62	75	6E	74	75	20	39	2E	33	2E	30	2D	31	37	75	62	75	GCC: (Ubuntu 9.3.0-17ubu
00000120	6E	74	75	31	7E	32	30	2E	30	34	29	20	39	2E	33	2E	30	00	00	00	00	00	00	00	ntul-20.04) 9.3.0.....
00000138	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000150	00	00	00	00	03	00	01	00	18	03	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000168	00	00	00	00	03	00	02	00	38	03	00	00	00	00	00	00	00	00	00	00	00	00	00	008.....

Signed 8 bit: 32 Signed 32 bit: 543255660 Hexadecimal: 20 61 6C 6C
 Unsigned 8 bit: 32 Unsigned 32 bit: 543255660 Decimal: 032 097 108 108
 Signed 16 bit: 8289 Float 32 bit: 1.909412E-19 Octal: 040 141 154 154
 Unsigned 16 bit: 8289 Float 64 bit: 1.03960468256775E-152 Binary: 00100000 01100001 01101100 011011
☐ Show little endian decoding ☐ Show unsigned as hexadecimal ASCII Text: all
 Offset: 96 / 4719 Selection: 0 to 95 (96 bytes) INS

2nd p 128 byte part. 224 byte point. After that, the remains are final segments of suffix.



File Edit View Search Tools Help

suffix

00000000	20	77	68	65	72	65	20	69	74	20	69	73	28	77	68	69	63	68	20	65	76	65	72	20	where it is(which ever
00000018	69	73	20	67	72	65	61	74	65	72	29	20	69	74	2E	2E	00	00	00	00	00	00	00	00	is greater) it.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	54	68	65	20	6D	69	73	73The miss
00000048	69	6C	65	20	6B	6E	6F	77	73	20	77	68	65	72	65	20	69	74	20	69	73	20	61	74	ile knows where it is at
00000060	20	61	6C	6C	20	74	69	6D	65	73	2E	20	49	74	20	6B	6E	6F	77	73	20	74	68	69	all times. it knows thi
00000078	73	20	62	65	63	61	75	73	65	20	69	74	20	6B	6E	6F	77	73	20	77	68	65	72	65	s because it knows where
00000090	20	69	74	20	69	73	6E	20	74	2E	20	42	79	20	73	75	62	74	72	61	63	74	69	6E	it isn t. By subtractin
000000a8	67	20	77	68	65	72	65	20	69	74	20	69	73	20	66	72	6F	6D	20	77	68	65	72	65	g where it is from where
000000c0	20	69	74	20	69	73	6E	20	74	20	6F	72	20	77	68	65	72	65	20	69	74	20	69	73	it isn t or where it is
000000d8	6E	20	74	20	66	72	6F	6D	20	77	68	65	72	65	20	69	74	20	69	73	28	77	68	69	n t from where it is(whi
000000f0	63	68	20	65	76	65	72	20	69	73	20	67	72	65	61	74	65	72	29	20	69	74	2E	2E	ch ever is greater) it..
00000108	47	43	43	3A	20	28	55	62	75	6E	74	75	20	39	2E	33	2E	30	2D	31	37	75	62	75	GCC: (Ubuntu 9.3.0-17ubu
00000120	6E	74	75	31	7E	32	30	2E	30	34	29	20	39	2E	33	2E	30	00	00	00	00	00	00	00	ntul-20.04) 9.3.0.....
00000138	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000150	00	00	00	00	03	00	01	00	18	03	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000168	00	00	00	00	03	00	02	00	38	03	00	00	00	00	00	00	00	00	00	00	00	00	00	008.....

Signed 8 bit: 32 Signed 32 bit: 544696421 Hexadecimal: 20 77 68 65
 Unsigned 8 bit: 32 Unsigned 32 bit: 544696421 Decimal: 032 119 104 101
 Signed 16 bit: 8311 Float 32 bit: 2.095626E-19 Octal: 040 167 150 145
 Unsigned 16 bit: 8311 Float 64 bit: 2.79333332678735E-152 Binary: 00100000 01110111 01101000 01100
☐ Show little endian decoding ☐ Show unsigned as hexadecimal ASCII Text: whe
 Offset: 224 / 4719 Selection: 96 to 223 (128 bytes) INS

After cutting pieces of those files, concatenate prefix + p/q + 1st segment of suffix + p + rest of suffix will lead to final_1 and final_2 program. Final_1 has both p's, and Final_2 has p and q. Run them, it will be shown that even though the p/q part has changed, as long as rest of the binary files are not tampered, the outcome is still same. (Run benign code).