

# BINARY EXPLOITATION

MADE WITH ❤ BY WRTH

C add.c 1 X



C add.c

*Open chat (Ctrl+I), or select a Language (Ctrl+K M), or fill with template to get started.*

*Start typing to dismiss or don't show this again.*

I

Buatlah program C yang menginput 2 angka dan mengoutputkan hasilnya

Berikut adalah contoh program C sederhana yang menginput dua angka dari pengguna, kemudian menghitung dan menampilkan hasil penjumlahan, pengurangan, perkalian, dan pembagian:

```
c
```

Copy Edit

Ask anything



Search

Reason

Deep research

Create image

...



C add.c X

▷ ⌂ ⚙ ⏺ ⋮

C add.c > ...

```
#include <stdio.h>

int main(){
    int angka1;
    int angka2;
    scanf("%d %d", &angka1, &angka2);
    int angka3 = angka1 + angka2;
    printf("%d\n", angka3);
}
```

Runable code for  
int main(){  
int angka1;  
int angka2;  
scanf("%d %d", &angka1, &angka2);  
int angka3 = angka1 + angka2;  
printf("%d\n", angka3);  
}

C add.c X



C add.c > ...

```
#include <stdio.h>

int main(){
    int angka1;
    int angka2;
    scanf("%d %d", &angka1, &angka2);
    int angka3 = angka1 + angka2;
    printf("%d\n", angka3);
}
```

Available variables:

```
int main();
int angka1;
int angka2;
int angka3 = angka1 + angka2;
printf("%d\n", angka3);
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

sh - laos + ▾ □ ⌂ ⌃ ⌄ X

```
$ ./calc
10 9
19
$ ./calc
111111111111 111111111111
-1116077170
$ 
```

C add.c X

```
#include <stdio.h>
```

# Convincing

```
int main(){
    int angka1;
    int angka2;
    scanf("%d %d", &angka1, &angka2);
    int angka3 = angka1 + angka2;
    printf("%d\n", angka3);
```

# Computers

To  
\$ ./cal

-1116077170

# Themselves



# Computers are weird

They're not like humans

You need to be **specific**

Not being specific creates all sorts of  
problems

# Integer (int)

Stores numbers

```
#include <stdio.h>

int main(){
    int angka1;
    int angka2;
    scanf("%d %d", &angka1,
    int angka3 = angka1 + an
    printf("%d\n", angka3);

}
```

```
#include <stdio.h>

int main() {
    // Integer types
    short s = 1;
    unsigned short us = 1;
    int i = 1;
    unsigned int ui = 1;
    Long l = 1;
    unsigned Long ul = 1;
    Long Long ll = 1;
    unsigned Long Long ull = 1;
    float f = 1;
    double d = 1;
    Long double ld = 1;
}
```

# Be Specific!

There are 11 ways to store a number in C

# Integer

32 Bit

0 0

# Integer

1 + 31 Bit

0 0

# Integer

# 1 + 31 Bit

# Sign Bit

Indicates the sign  
of a number



0 0

# Integer

## 1 + 31 Bit

**0 1 0 1 1 1 0 1 0 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1**

2

**1564391411**

\*Technically this is NOT the accurate visualization of integers but this is good enough for now I'm not gonna bother you with too much explanation, it's a thing we like to call "something we're not gonna get into"

# Integer

1 + 31 Bit

**1 1 0 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1**

=

**-1564391411**

\*Technically this is NOT the accurate visualization of integers but this is good enough for now I'm not gonna bother you with too much explanation, it's a thing we like to call "something we're not gonna get into"

# Integer

The Problem

0	1	0	1	1	1	0	1	0	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	0	1	1	
0	1	0	1	1	1	0	1	0	0	1	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1	1	0	0	1	1
																									+ +					
1	0	1	1	1	0	1	0	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	0	0	1	1	



R

# New Bug Unlocked!

Occurs when an arithmetic operation on integers attempts to create a numeric value that is outside of the range that can be represented with a given number of digits

JULIA EVANS  
@bork

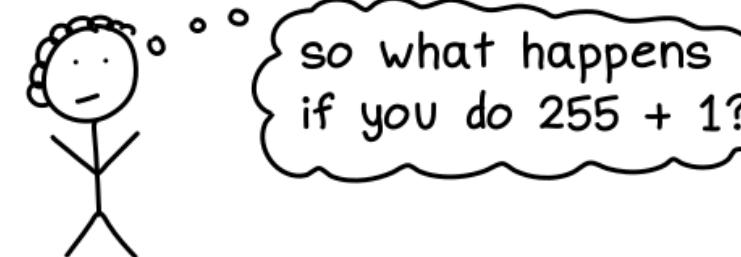
## integer overflow

integers have a limited amount of space

The usual sizes for integers are:

8 bits: 32 bits = 4 bytes  
16 bits:   
32 bits: 64 bits:   
64 bits is often the default these days.

the biggest 8-bit unsigned integer is 255



Going above/below the limits is called overflow.

ways overflow is handled

① wrap around

$$255 + 1 = 0$$

$$255 + 3 = 2$$

② raise an error

③ saturate ← this one is unusual

$$255 + 1 = 255$$

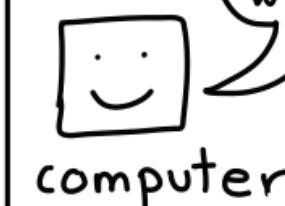
$$255 + 3 = 255$$

maximum numbers for different sizes

bits	signed	unsigned
8	127	255
16	32767	65535
32	~2 billion	~4 billion
64	~9 quintillion	~18 quintillion

overflows often don't throw errors

255 + 1? that number is 8 bits, so the answer is 0! that's what you wanted right?



This can cause VERY tricky bugs.

some languages where integer overflow happens

Java/Kotlin C/C++ Rust  
SQL C# Swift Go R  
Dart

Some throw errors on overflow, some don't, for some it depends on various factors. Look up how it works in your language!

# **Simple Case**

# Water Break

**PRACTICE THIS EXERCISE**



```
#include <stdio.h>

int main(){
    char name[8];
    printf("Your name: ");
    scanf("%s", name);
    printf("Hello %s, Mada
kono sekai wa", name);
}
```

# Characters (char)

Stores characters

TECHNICALLY it can also stores integers, since everything is binary anyways



[https://www.reddit.com/r/linuxmemes/comments/1guythg/me\\_after\\_using\\_a\\_diy\\_distro\\_for\\_the\\_first\\_time/](https://www.reddit.com/r/linuxmemes/comments/1guythg/me_after_using_a_diy_distro_for_the_first_time/)

# ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

# Strings

N + 1 bytes

6f 70 65 6e 00 70 61 69 64 00 70  
72 6f 6d 6f 74 65 00 64 6d 00

# Strings

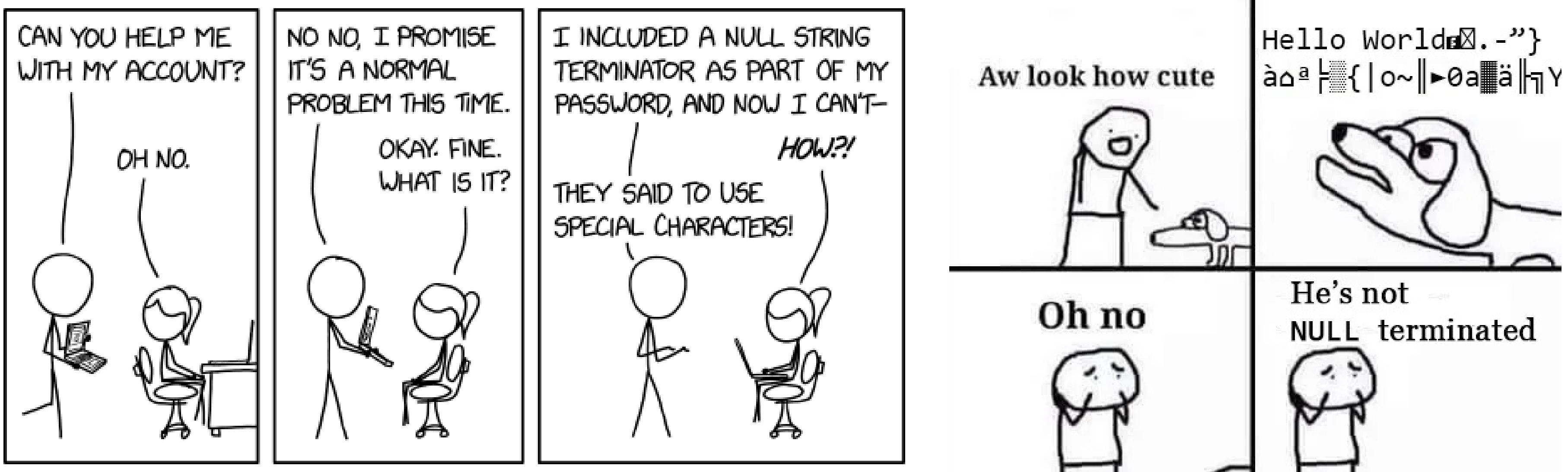
The Problem

-\\_(ツ)\_/-

# New Bug Unlocked!

# Null Terminated Strings issue

This is usually not a problem, because normal text never contains null characters. However, if somehow a null character were to end up in the string, it would cause problems: any code that uses that string would assume this null character marks the end of the string, so the string would effectively be cut off.



XKCD 2700

[https://www.reddit.com/r/ProgrammerHumor/comments/8win5c/please\\_train\\_your\\_dog\\_good\\_behavior/](https://www.reddit.com/r/ProgrammerHumor/comments/8win5c/please_train_your_dog_good_behavior/)

# New Bug Unlocked!

## Buffer Overflow (Basic)

A buffer overflow or buffer overrun is an anomaly whereby a program writes data to a buffer beyond the buffer's allocated memory, overwriting adjacent memory locations.



" A BUFFER OVERFLOW, OCCURS WHEN MORE DATA IS PUT INTO A FIXED-LENGTH BUFFER THAN THE BUFFER CAN HANDLE. "

# New Bug Variant Unlocked!

## Buffer Overflow, Off By One Variant

A product calculates or uses an incorrect maximum or minimum value that is 1 more, or 1 less, than the correct value. In this case an input function is taking 1 more bytes than it should



"A BUFFER OVERFLOW, OCCURS WHEN MORE DATA IS PUT INTO A FIXED-LENGTH BUFFER THAN THE BUFFER CAN HANDLE."

# Simple Case

# Water Break

**PRACTICE THIS EXERCISE**



```
#include <stdio.h>

int main(){
    int a[8] = {1,2,3,4,5,6,7,8};
    for(int i = 0; i < 8; i++){
        printf("%d\n", a[i]);
    }
}
```

# Array

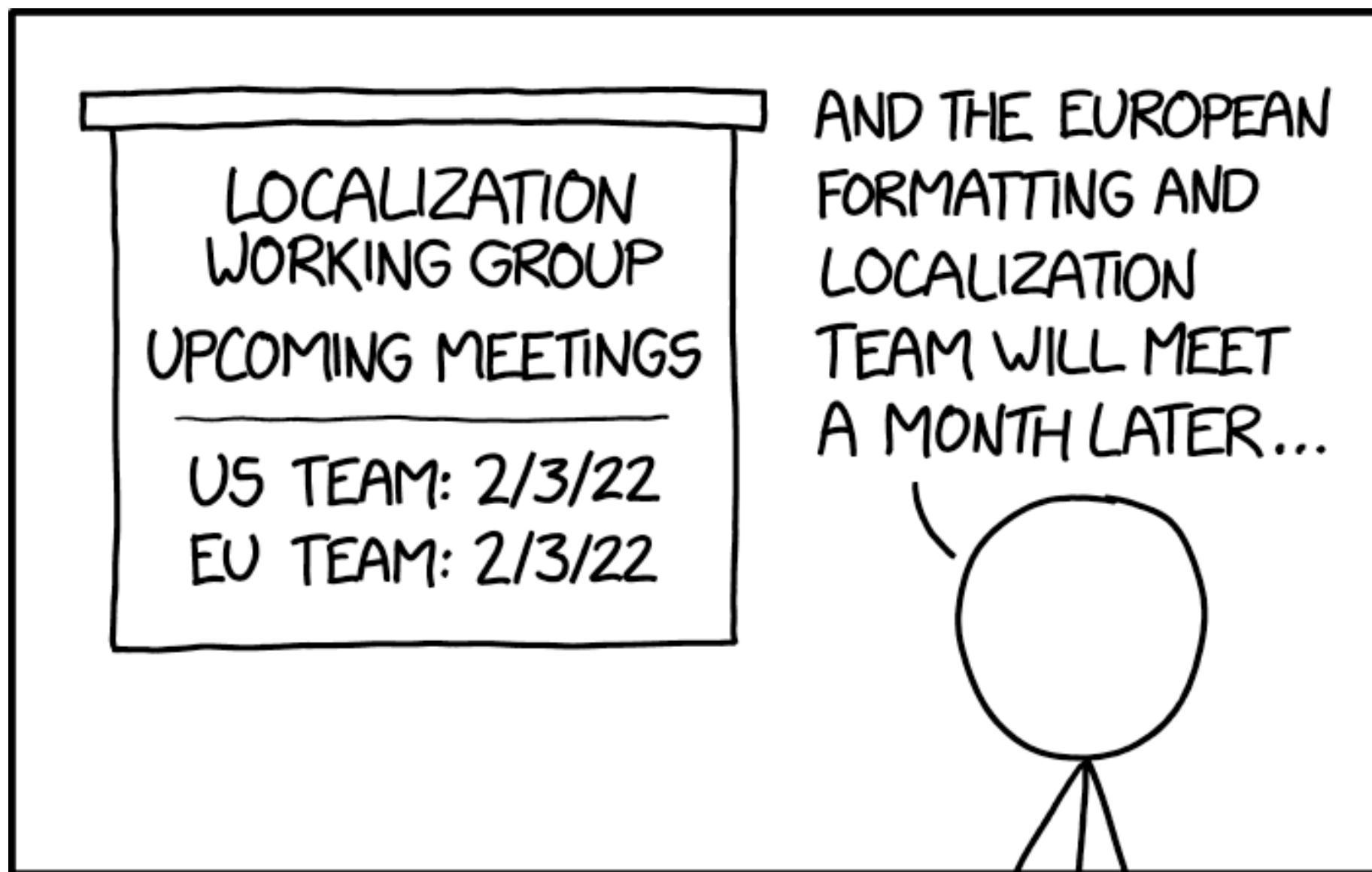
Stores multiple <types>

Accessed using what's called an index  
Index starts at 0

# New Concept Learned!

## Endianness

the order in which bytes are arranged within a multi-byte data unit, like an integer or a floating-point number, when stored in memory or transmitted over a network

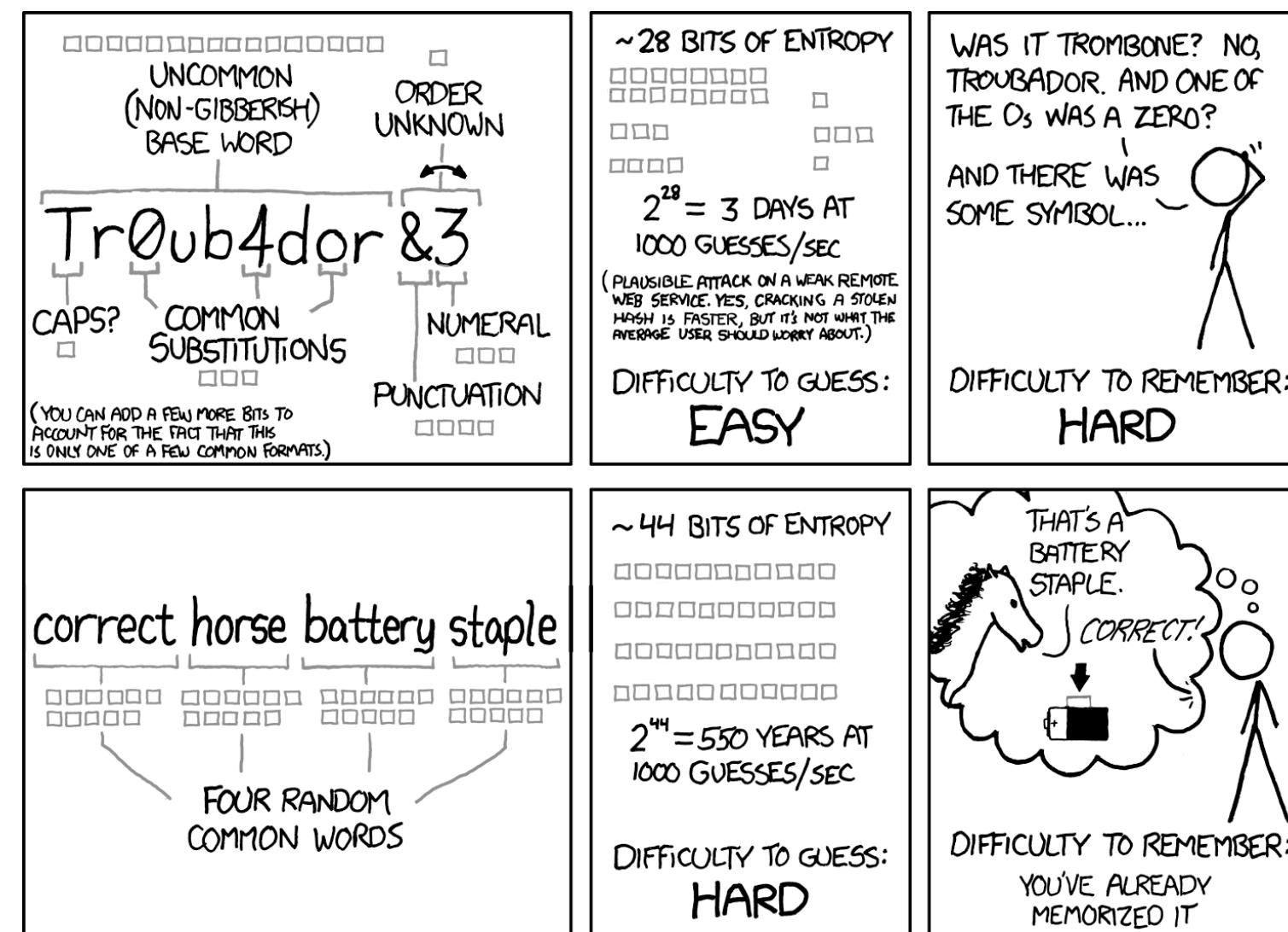


XKCD 2562

# New Bug Unlocked!

## Out Of Bounds (OOB)

An out-of-bounds happens when a program tries to read (or write) data from a memory location that's outside the allocated buffer



XKCD 936

# **Simple Case?**

# CHAPTER 1 COMPLETE!

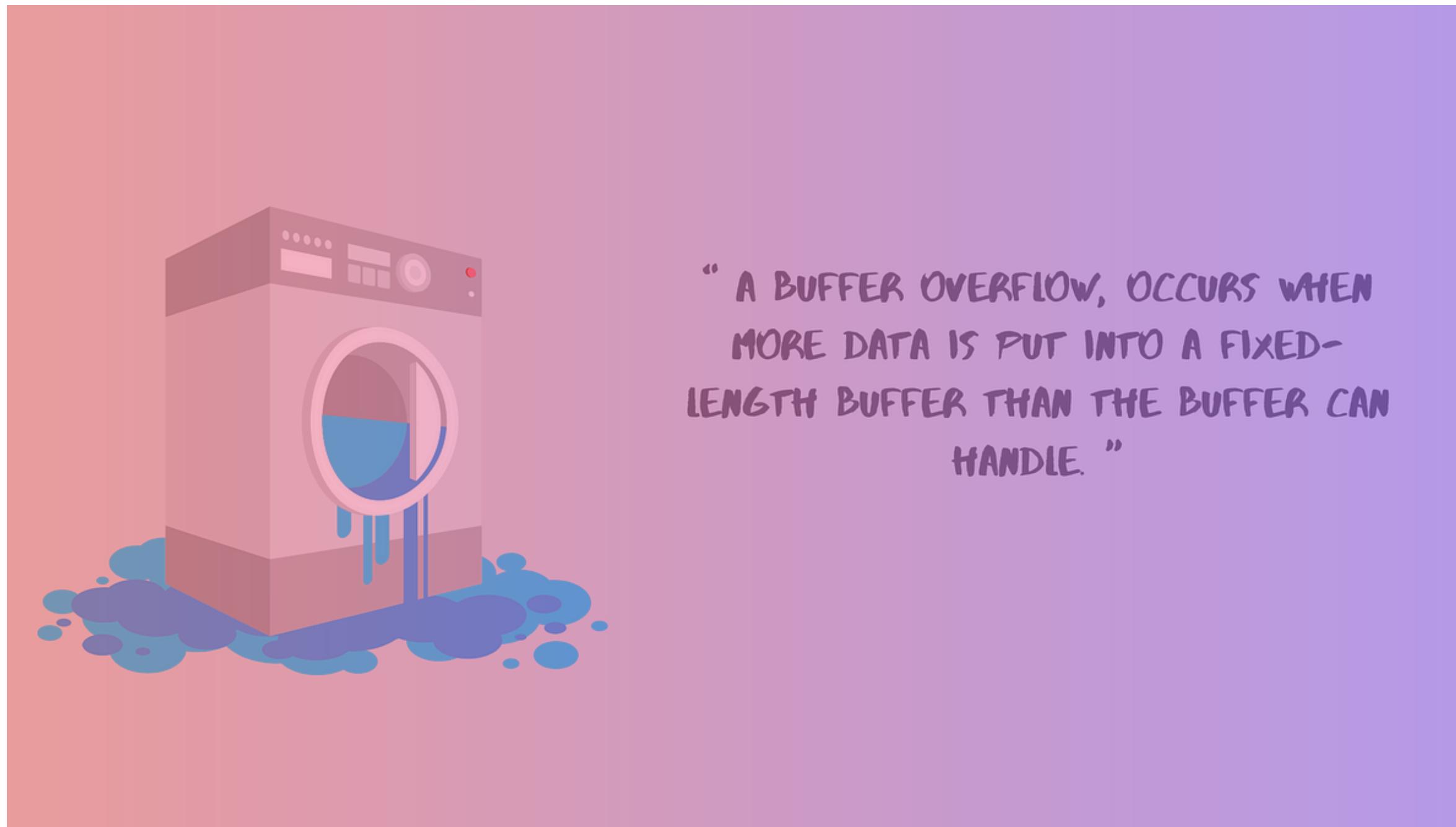
**PRACTICE THIS EXERCISE**



# New Bug Unlocked!

## Buffer Overflow (Advanced)

A buffer overflow or buffer overrun is an anomaly whereby a program writes data to a buffer beyond the buffer's allocated memory, overwriting return address and thus controlling the program's flow.



# New Concept Learned!

## Shellcode

the order in which bytes are arranged within a multi-byte data unit, like an integer or a floating-point number, when stored in memory or transmitted over a network



XKCD 1247

# New Bug Unlocked!

## Format String Bug (FSB)

the use of unchecked user input as the format string parameter in certain C functions that perform formatting, such as printf()

Shantonu Sen  
@shantonusen

My kids just asked why there was a Minecraft update with no features and what a “Log4J” was, and I have been preparing my whole life for this.

I had to start at the beginning with C format strings. I should be able to get to Java and jar files by midnight.

4:57 AM · Dec 12, 2021 · Twitter for iPhone

# See You Next Campaign!

PRESENTED WITH ❤ BY WRTH