DEF CON 27
Capture the Flag *Finals*

Shortman

The CTF

Live Attack/Defense CTF

16 Teams from all over the world

Must qualify by either winning a qualifier or finishing in the top X in the Defcon qualifier CTF

Pre-qualified Teams

DEF CON 2018 CTF - 12 August 2018 - prequalified: DEFKOR00T

HITCON CTF 2018 - 21 October 2018 - prequalified: Dragon Sector

RuCTFE 2018 - 10 November 2018 - prequalified: saarsec

C3CTF 2018 - 27 December 2018 - prequalified: mhackeroni

PlaidCTF 2019 - 12 April 2019 - prequalified: HITCON

Defcon Qualifiers

#	Team	Completed	Speedrun Individual	Speedrun Overall	Points
1	PPP	👽👽👽👽🖐🚀🚀🚀🔫🖐🔫🗿🗿🗿🔫🗿🔫🔫🚀🚀🖐🔫	90	0	3681
2	HITCON⚔BFKinesiS	👽👽👽👽🖐🗿🚀🚀🚀🔫🗿🗿🔫🖐🔫🗿🖐🗿🚀🔫🚀🔫	100	100	3571
3	SeoulPlusBadAss	👽👽👽🚀👽🖐🚀🚀🚀🗿🔫🔫🔫🗿🗿🖐🗿🚀🔫	235	300	3230
4	A*0*E	👽👽👽👽🖐🚀🚀🗿🚀🔫🖐🔫🗿🗿🖐🔫🗿🚀🔫🚀	135	200	3195
5	Shellphish	👽👽👽👽🖐🚀🚀🗿🔫🔫🚀🔫🗿🚀🖐🚀🗿🗿🔫	65	0	3100
6	Sauercloud	👽👽👽🚀👽🚀🗿🖐🖐🔫🔫🗿🖐🔫🚀🗿🚀🗿🔫🚀	55	0	2932
7	Samurai	👽👽👽🚀🖐🚀👽🔫🚀🔫🗿🖐🔫🗿🖐🔫🚀🚀🔫	55	0	2918
8	Tea Deliverers	👽👽👽🖐👽🚀🗿🔫🚀🗿🚀🗿🔫🗿🖐🚀🔫🔫🚀	75	0	2734
9	CGC	👽👽👽👽🖐🗿🚀🚀🗿🗿🗿🔫🔫🗿🚀🚀🔫🚀	100	0	2647
10	r00timentary	👽👽👽👽🖐🔫🚀🚀🔫🚀🗿🔫🔫🔫🗿🚀🚀	150	0	2628
11	hxp	👽👽👽🚀🚀👽🖐🗿🖐🗿🔫🗿🚀🔫🚀🖐🗿🗿	55	0	2602
12	KaisHack GoN	👽👽👽👽🚀🚀🗿🔫🗿🗿🔫🔫🚀🚀🖐🗿🚀🖐	60	0	2465
13	TokyoWesterns	👽👽👽🖐🚀🚀👽🖐🗿🗿🗿🔫🖐🗿🗿🔫🚀	60	0	2401
14	r3kapig	👽👽👽👽🖐🚀🚀🚀🗿🗿🔫🗿🚀🖐🗿🚀🖐	60	0	2337
15	RPISEC	👽👽👽👽🚀🗿🔫🔫🗿🗿🚀🚀🚀🚀🗿🖐🖐	65	0	2283

Thursday (Day -1)

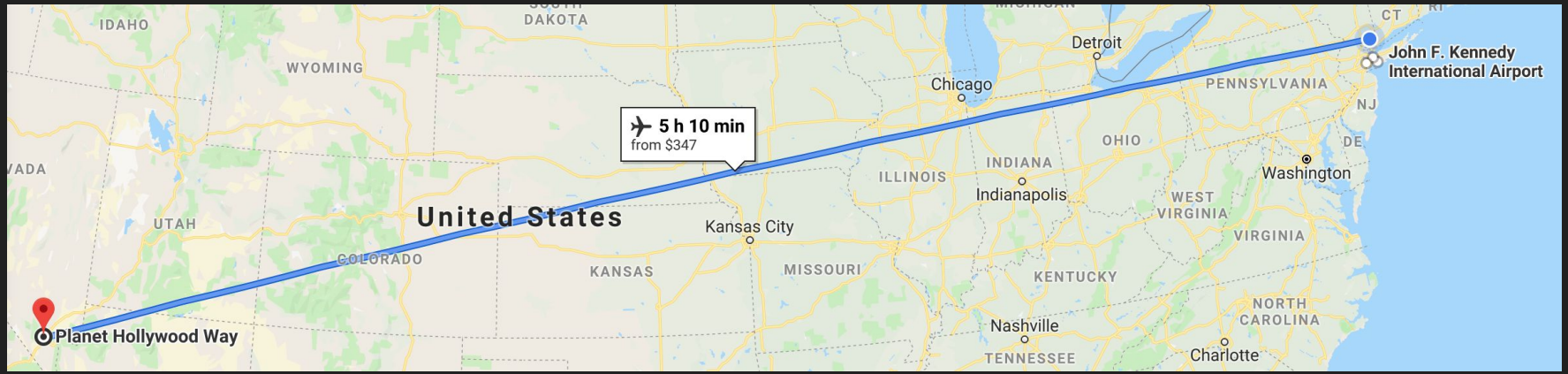
We get an information “leak” from the Order of the Overflow, that instructed us to bring the following tools:

- Microsoft Windows + Visual Studio
- MacOS + XCode + iOS SDK
- Any GNU/Linux distribution with proper toolchain + Android SDK
- FreeBSD (comes with toolchain)

- An extra monitor that supports HDMI...

Thursday (Day -1)

Arrived at 12:30am after delayed flight from JFK to Planet Hollywood



Friday (Day 1)

Game started at 10am (after ~5 hours of sleep)

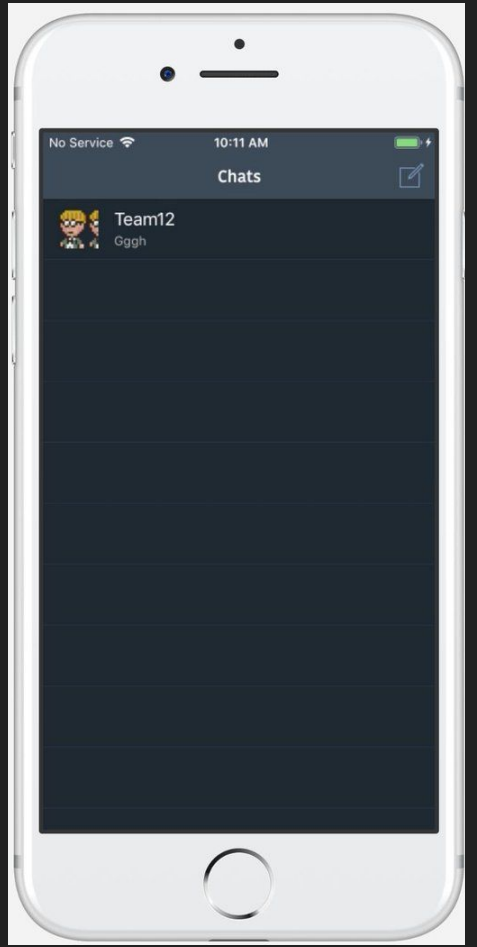
First challenges released:

- TelOoOgram: iOS messaging app similar to telegram (Objective C)
- AoOoL: Webserver, written in ??
- ROPShip: King of the Hill challenge

Hackers Don’t Use Macs….

But I actually brought my UCSB Macbook Pro

Hello TeloOogram!



TeloOogram

- First bug identified
  - Unused “VoIP” server with a trivial buffer overflow
  - Appeared to be unexploitable
  - Easily patched (patch deployed)

TeloOogram

- Second bug identified
  - The app requests avatar.png from contacts
  - Let’s try requesting other files…
  - Success. Stole other teams creds.txt (username/password)
  - Oh yeah, and their flags
  - Easily patched (patch deployed)
- Saarsec getting more flags that us, but not exploiting us…
  - Hours pass…
  - Turns out other teams aren’t great at patching
    - Try **./flag** instead of **flag**

TeloOogram

- Third bug identified
  - Objective C parser used that was deprecated for security reasons
  - This is a nasty one…
  - Goes unexploited by any team, despite our best efforts
  -

TeloOogram

- Removed from the game at the end of Day 1
  - We rejoice

AoOol

Some webserver written in C/C++

- Responds to GET, UPLOAD, and CONFIG commands

Looks like there are some funky bits with parsing of a config file

I start getting spun up… then fall asleep.

Saturday (Day 2)

Game starts at 10am (again)

- Actually a little bit late, but that’s normal
- I start working on AoOol again, until...

fish we are getting a team XBox

be ready!!!!

rhelmot 🏳️‍🌈 A fu n ing what

Okay I guess I'm coming to the floor

👍 2

fish we are getting a team XBox

be ready!!!!

rhelmot 🏳️‍🌈 A fu[redacted]ing what

Okay I guess I'm coming to the floor

👍 2

shortman 🤘 I used to mod xboxes as a side business

fish we are getting a team XBox

be read

rhelmot

Okay I

2

sl

salls @shortman you should come here

to work on the xbox stuff

to the floor

shortman Is there a seat?

salls yeah

one of us will switch

guys we have an issue with the xbox, anyone expert at networking?

zanardi xbox experts should come to the floor now

however many

degrigis @shortman is coming

DoOom on an original XBOX

DoOom on an original XBOX

First, The Good

The XBOX had been modded to download a .xbe file over the network

It was downloading a version of Chocolate Doom

Multiplayer game against other teams!

Scoring:

- Find OOO tiles and stand on them (1 point per second)

The hard stuff

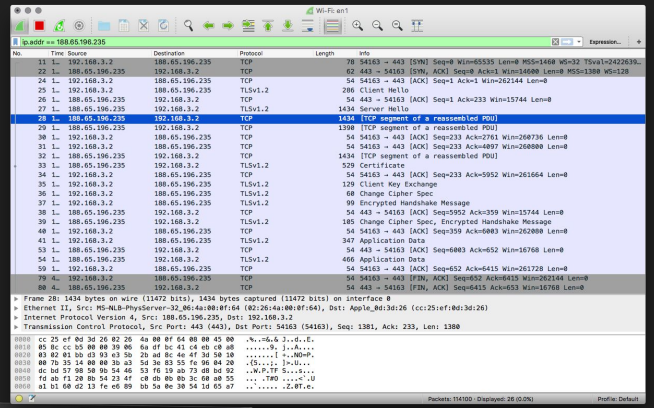
We are told that the XBOX must be “pingable” (turns out to be a lie…)

The original .xbe has shooting disable and username “sheeple”

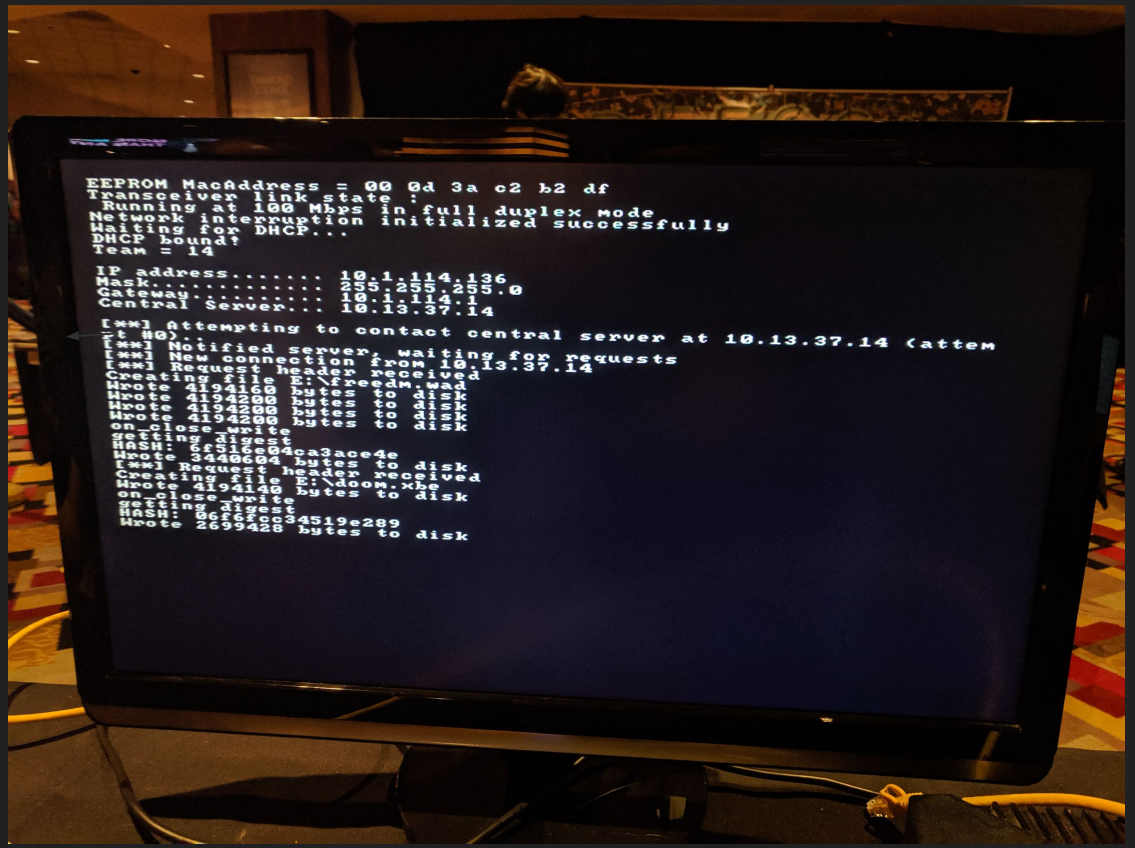
You can only score with the username of your team id

E.g., [14]shellphish

Let the pwning begin!



Let the pwning begin!



Let the pwning begin!

Shooting enabled, points being scored… but… there's more..

WE FIND A HIDDEN ROOM THAT IS COVERED IN OOO TILES

The catch: you need to clip through walls to get there

Becoming a God

We patch the binary to enable no clipping

IT WORKS!

We freak!

Becoming a God

No points are being scored…

- Actually we can’t tell if points are being scored

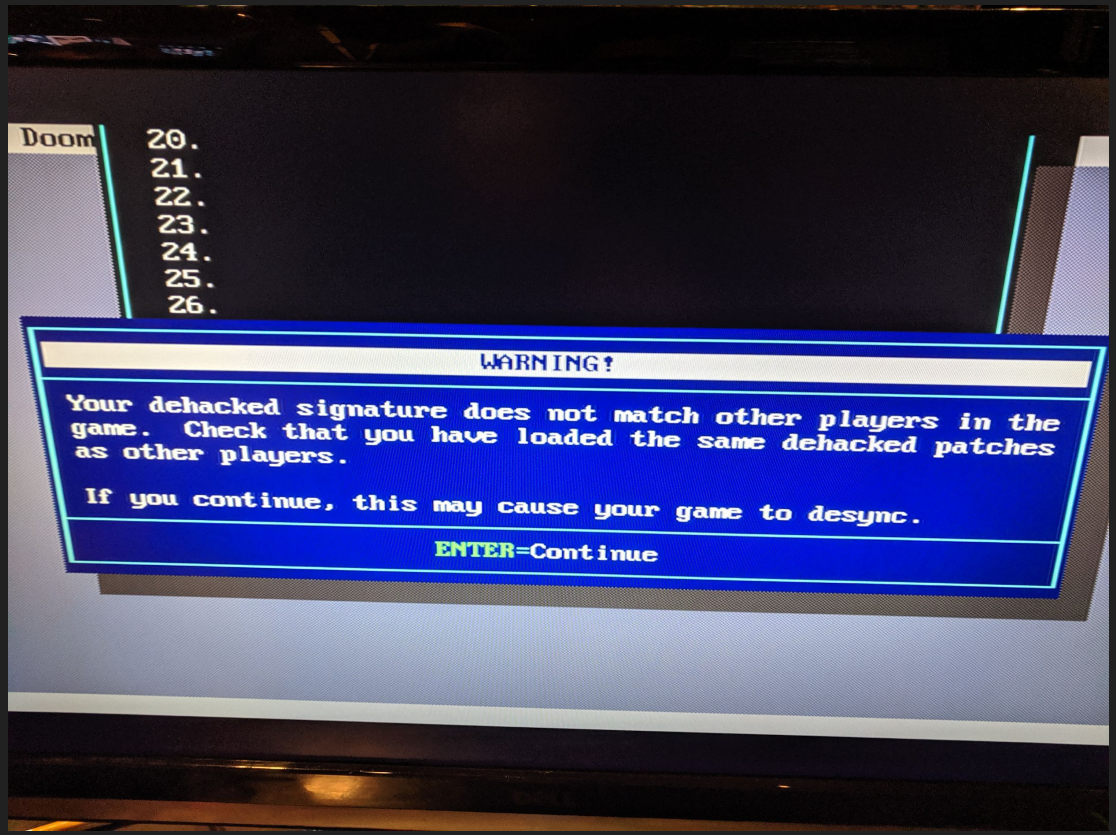
OOO tells us everything is fine

We fight for hours..

We don’t know if it’s working, or if we are scoring,

but we are Gods.

We were DoOomed



We were DoOomed

We needed to send our commands to the server as well, not just locally patch…

Also, the XBOX didn’t need to be pingable…

Lack of feedback killed us.

We complained to the organizers, they promised to fix it next year.

End of Friday

Finally, some rest…

What are the other challenges?

The Bitflip Conjecture

==========

Definition:

A snippet of assembly code is `N-Flip Resistant` if its output remains constant (i.e., it produces the same output and exits with the same return value) even if ANY combination of N bits are flipped.

One-flip Conjecture:

The x86 architecture is such that it is possible to write any arbitrary program (of any length) in a way that is 1-flip resistant.

- Balzaroth (Vegas 2019)

The Bitflip Conjecture

Points are assigned based on how close you are from a complete proof

(i.e., based on how many bit flip your code was able to withstand)

But first, how do you want the registers initialized before executing the code?

1. I like all my registers set to zero
2. I want them pointing to the middle of a 64KB R/W region of memory)
3. Dont bother. Leave them as they are

The Bitflip Conjecture

We are allotted 200 bytes of shellcode

This happens to be closely related to my research here…

Game on!

The Bitflip Conjecture

Actually, the CTF is paused so we can’t score

But we can still get our shellcode ready for morning

The Bitflip Conjecture: Idea 1

Replicate shellcode, and do a checksum

```
   BITS 64

_start:
        lea rax, [rel copy2]
        lea rbx, [rax-(copy2 - copy1)]
loop_start:
        dec al
        add cl, byte [rax]      ; add cl, [rax]
        cmp eax, ebx
        jnz loop_start

decide:
        cmp cl, 34
        jnz copy2

copy1:
        db SHELLCODE

copy2:
        db SHELLCODE
```

The Bitflip Conjecture: Idea 1

Replicate shellcode, and do a checksum

```
[--xxxxxx]  [xxxxxxxx]  [xxxxxxxx]  [--------]  [------xx]  [xxxxxxxx]  [xxxxxxxx]  [---xxxxx]
[-------x]  [x-xxxxxx]  [xxxxxxx-]  [xxxx-xxx]  [---xx-xx]  [-xxx---x]  [------x-]  [-xxx-x-x]
[--x-xxx-]  [--x-xxx-]  [------xx]  [-----x-x]  [--xxxxxx]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [xxxxxxxx]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]  [--------]
[--------]  [--------]  [--------]  [--------]  [--------]
```

The Bitflip Conjecture: Idea 2

Transactional Memory!

If the transaction fails, it will reset everything

PROBLEM 1: The xbegin instruction will always fail bitflips

PROBLEM 2: We need to flush the instruction cache… cpuid fails too

Still… Pretty good (~12 bits)

The Bitflip Conjecture: Idea 3

What if we just fix the flipped bit…?

RAX = ptr to shellcode

RCX = offset to byte that was flipped

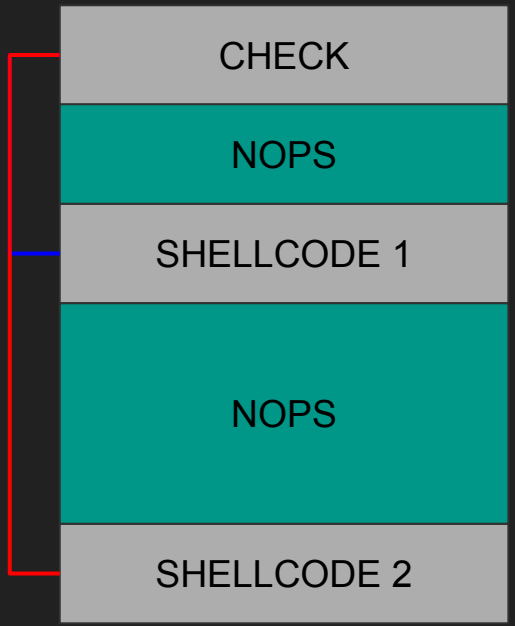
The bit that was flipped is on the stack somewhere

The Bitflip Conjecture: Idea 3 (Improved)

Check offset

Jump to uncorrupted portion of the code

Now only our check needs to survive bit flips...



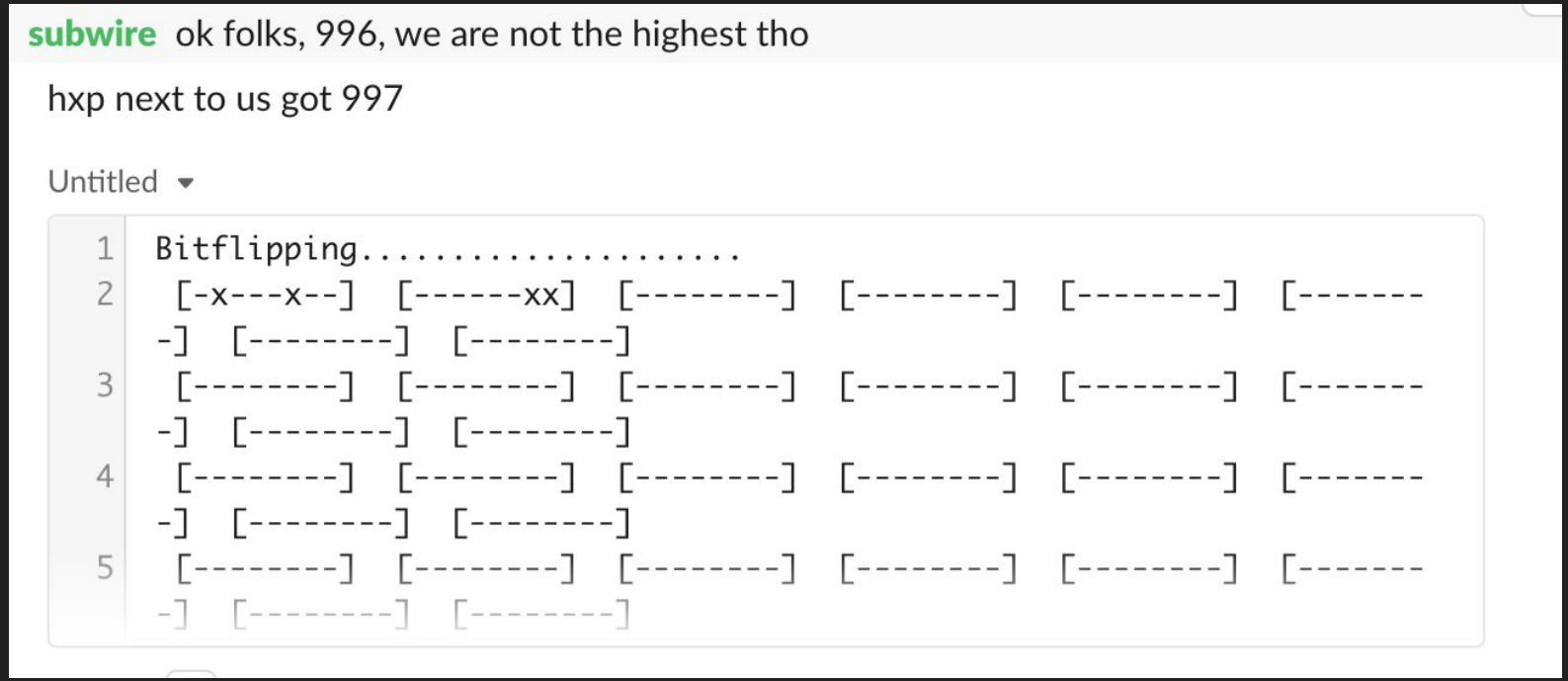
The Bitflip Conjecture: Idea 3 (Improved)

4 Bits!!!

```
BITS 64

_start:
        sbb cl, (0x22 + copy2)
        jbe $+0x67
post_jump:

copy1:
        db SHELLCODE

buf:
        times (64 - (buf - post_jump)) db 0x90

copy2:
        db SHELLCODE
```

Good, but not good enough

0 points scored



Good, but not good enough

fish announcement: if you want to score points for bitflip, you need to score more than or equal to 999...

@channel ^^^

saagarjha So someone has a perfect?!

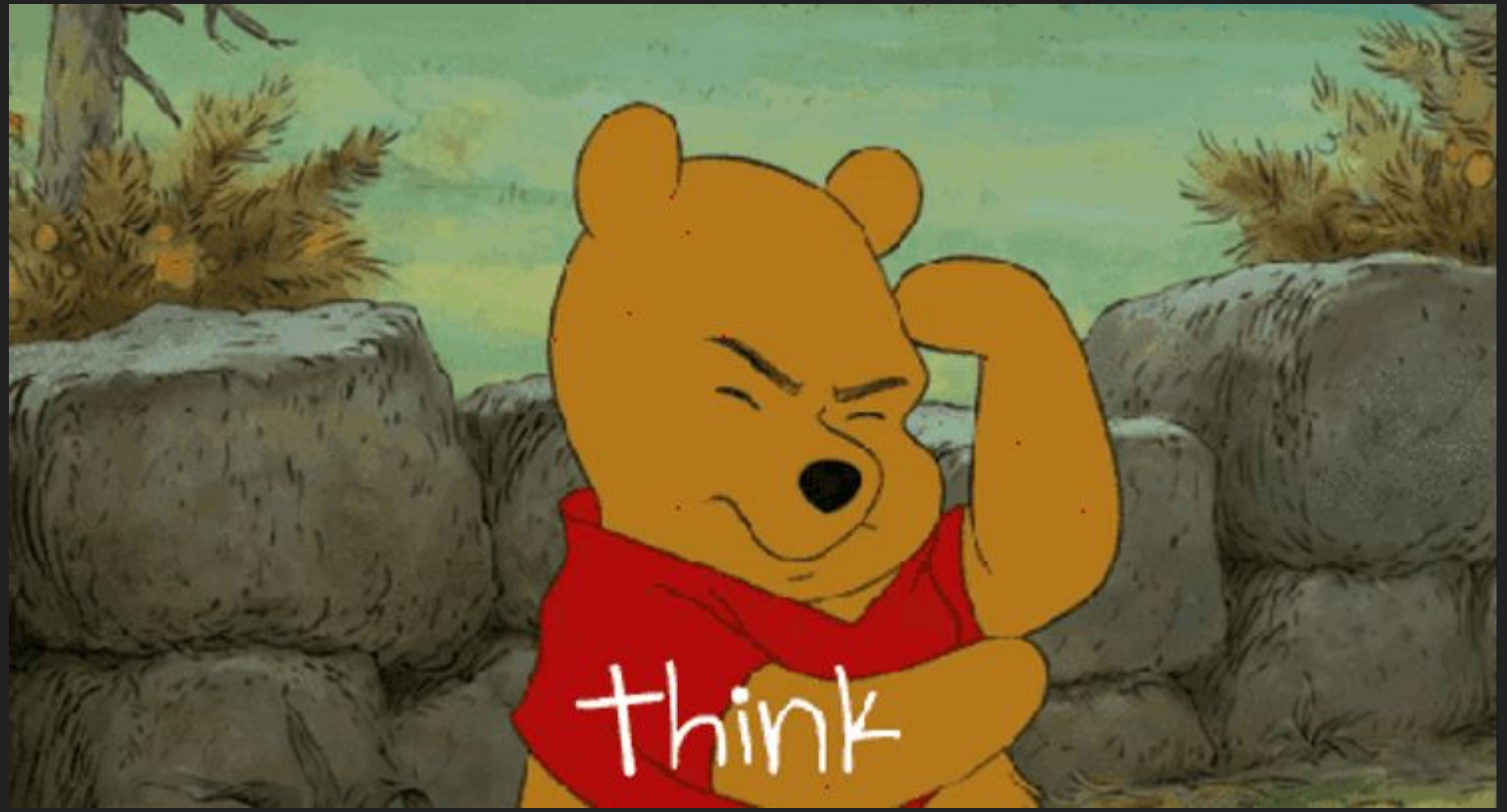
GH0S1 How much are we getting now?

saagarjha 996

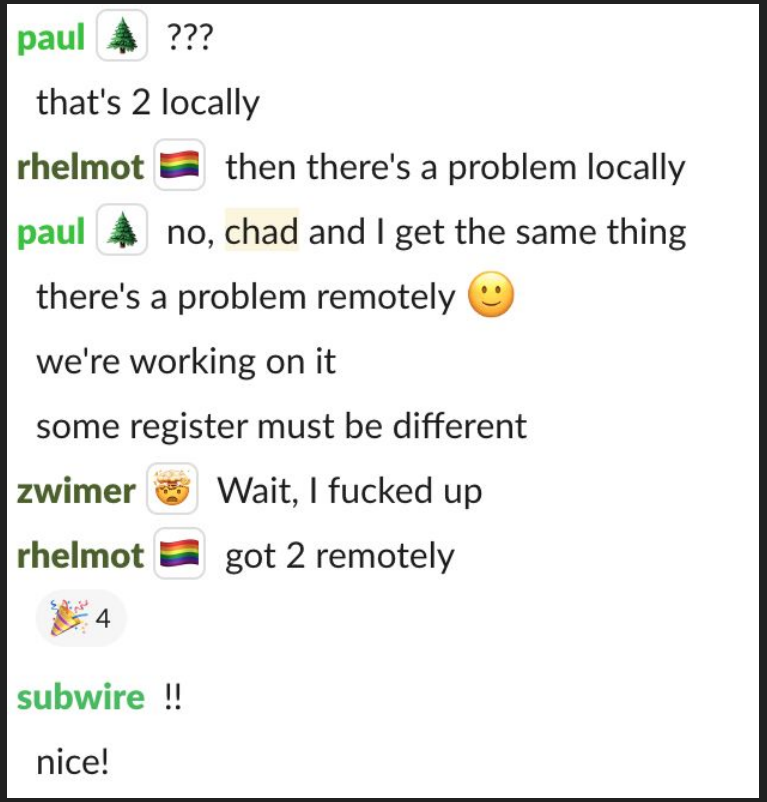
salls fu nnnn!

fish I bet 999 is 1-bit flipping

Good, but not good enough

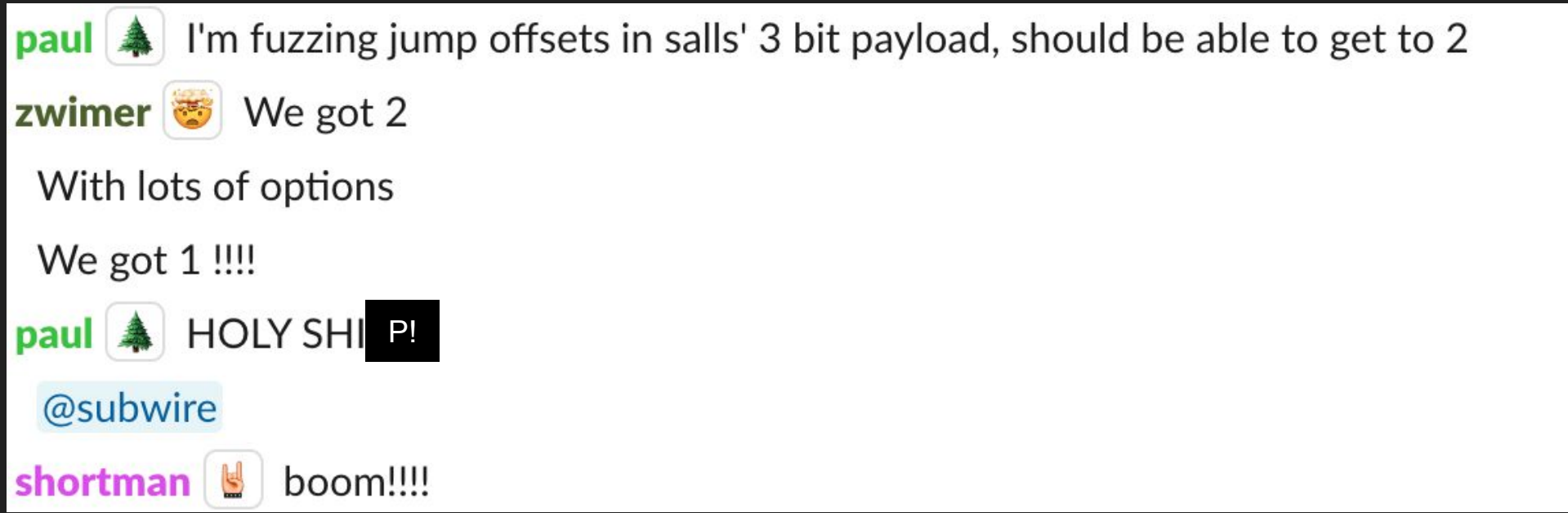


We can do better



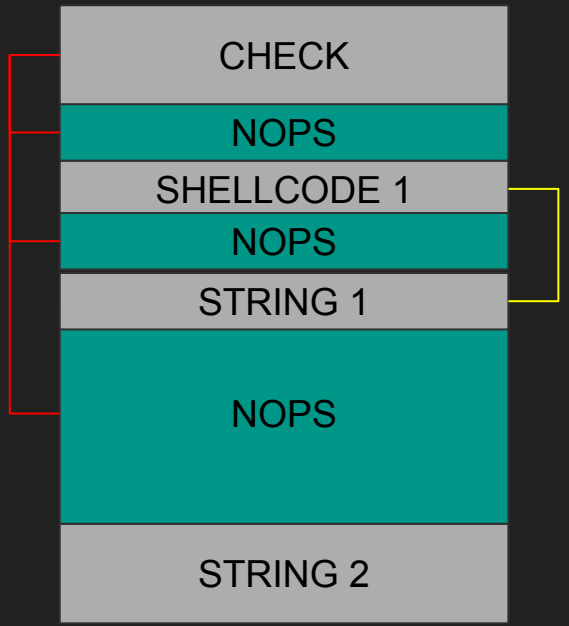
n

Let’s just fuzz offsets



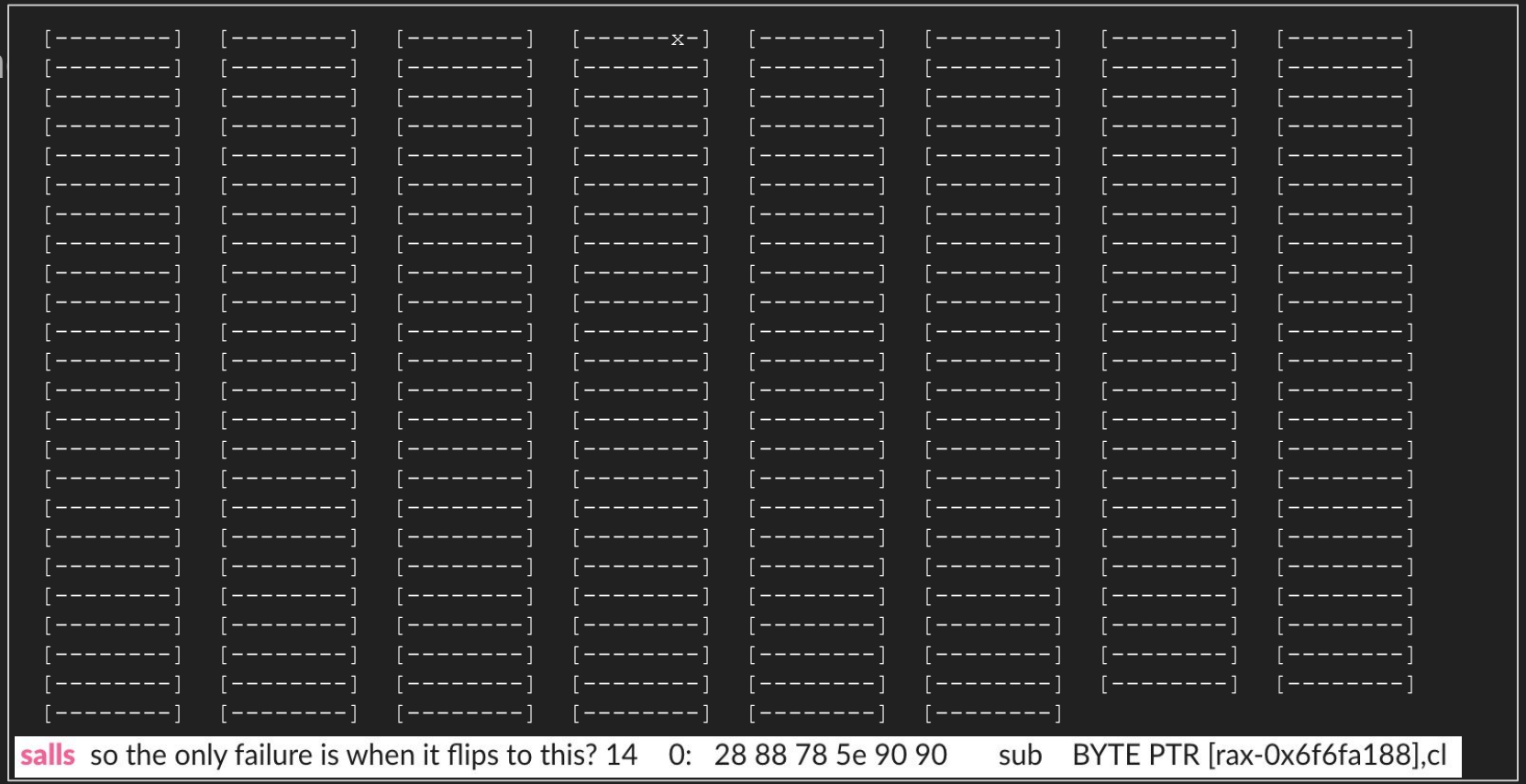
1 Bit!!!

```
        BITS 64
_start:
        add al, cl
        jns $+0x60
copy1:
        NOPS
        SHELLCODE
        NOPS
        jmp copy1
the_string1:
        db "I am Invincible!"
buf:
        NOPS
Copy2:
       NOPS
       SHELLCODE
       STRING
```



1 Bit!!!

sh


salls so the only failure is when it flips to this? 14 0: 28 88 78 5e 90 90 sub BYTE PTR [rax-0x6f6fa188],cl

How to get 0

US

Tea Deliverers

Final Scores

	TOTAL		Attack		Defense		KoH
973	Plaid Parliament of Pwning	1442	Plaid Parliament of Pwning	213	Plaid Parliament of Pwning	769	HITCON⚔BFKinesiS
▲772	HITCON⚔BFKinesiS	1006	HITCON⚔BFKinesiS	159	A*0*E	664	Plaid Parliament of Pwning
▲590	Tea Deliverers	815	Tea Deliverers	156	HITCON⚔BFKinesiS	477	A*0*E
▼564	A*0*E	656	mhackeroni	147	mhackeroni	459	Tea Deliverers
▲556	mhackeroni	646	Samurai	132	r3kapig	443	KaisHack GoN
▼399	Samurai	510	A*0*E	130	Tea Deliverers	405	Sauercloud
▼375	Sauercloud	499	r00timentary	127	Sauercloud	377	mhackeroni
▲359	r00timentary	339	SeoulPlusBadAss	111	r00timentary	370	SeoulPlusBadAss
▼331	SeoulPlusBadAss	292	saarsec	100	Samurai	333	TokyoWesterns
▼284	Shellphish	131	r3kapig	98	Shellphish	269	Shellphish
284	r3kapig	114	Sauercloud	88	KaisHack GoN	173	saarsec
▼281	KaisHack GoN	109	Shellphish	75	SeoulPlusBadAss	123	Samurai
▼235	saarsec	106	CGC	58	saarsec	59	CGC
▼215	TokyoWesterns	96	TokyoWesterns	54	TokyoWesterns	46	r00timentary
▲110	CGC	8	hxp	35	CGC	5	hxp
67	hxp	2	KaisHack GoN	34	hxp	0	r3kapig
$400\,{}^{a}/_{M_a} + 400\,{}^{d}/_{M_d} + 200\,{}^{k}/_{M_k}$		Σ_{tick} *(1 for each stolen flag)* $M_a = max(\uparrow,100) = 1442$		Σ_{tick} *(1 if non-exploited AND there were exploits)* $M_d = max(\uparrow,100) = 213$		Σ_{tick} *(per-service point logic)* $M_k = max(\uparrow,100) = 769$	

10th Place!