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

<table>
<thead>
<tr>
<th>#</th>
<th>Team</th>
<th>Completed</th>
<th>Speedrun Individual</th>
<th>Speedrun Overall</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PPP</td>
<td>![Icon]</td>
<td>90</td>
<td>0</td>
<td>3681</td>
</tr>
<tr>
<td>2</td>
<td>HITCON × BFKinesisS</td>
<td>![Icon]</td>
<td>100</td>
<td>100</td>
<td>3571</td>
</tr>
<tr>
<td>3</td>
<td>SeoulPlusBadAss</td>
<td>![Icon]</td>
<td>235</td>
<td>300</td>
<td>3230</td>
</tr>
<tr>
<td>4</td>
<td>A^0^E</td>
<td>![Icon]</td>
<td>135</td>
<td>200</td>
<td>3195</td>
</tr>
<tr>
<td>5</td>
<td>Shellphish</td>
<td>![Icon]</td>
<td>65</td>
<td>0</td>
<td>3100</td>
</tr>
<tr>
<td>6</td>
<td>Sauerccloud</td>
<td>![Icon]</td>
<td>55</td>
<td>0</td>
<td>2932</td>
</tr>
<tr>
<td>7</td>
<td>Samurai</td>
<td>![Icon]</td>
<td>55</td>
<td>0</td>
<td>2918</td>
</tr>
<tr>
<td>8</td>
<td>Tea Deliverers</td>
<td>![Icon]</td>
<td>75</td>
<td>0</td>
<td>2734</td>
</tr>
<tr>
<td>9</td>
<td>CGC</td>
<td>![Icon]</td>
<td>100</td>
<td>0</td>
<td>2647</td>
</tr>
<tr>
<td>10</td>
<td>r0Otimentary</td>
<td>![Icon]</td>
<td>150</td>
<td>0</td>
<td>2628</td>
</tr>
<tr>
<td>11</td>
<td>hxp</td>
<td>![Icon]</td>
<td>55</td>
<td>0</td>
<td>2602</td>
</tr>
<tr>
<td>12</td>
<td>KaisHack GoN</td>
<td>![Icon]</td>
<td>60</td>
<td>0</td>
<td>2465</td>
</tr>
<tr>
<td>13</td>
<td>TokyoWesterns</td>
<td>![Icon]</td>
<td>60</td>
<td>0</td>
<td>2401</td>
</tr>
<tr>
<td>14</td>
<td>r3kapig</td>
<td>![Icon]</td>
<td>60</td>
<td>0</td>
<td>2337</td>
</tr>
<tr>
<td>15</td>
<td>RP1SEC</td>
<td>![Icon]</td>
<td>65</td>
<td>0</td>
<td>2283</td>
</tr>
</tbody>
</table>
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)
- 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 f**king what
Okay I guess I’m coming to the floor
👍 2
fish  we are getting a team XBox
be ready!!!!

rhelmot 🇬🇧 A fucking 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

salls @shortman you should come here to work on the xbox stuff to the floor

Okay I

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

```assembly
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
  jnz copy1

copy1:
  db SHELLCODE

copy2:
  db SHELLCODE
```
The Bitflip Conjecture: Idea 1

Replicate shellcode, and do a checksum

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
</tbody>
</table>
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

subwire  ok folks, 996, we are not the highest tho

hxp next to us got 997
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** full nnnn!

**fish** I bet 999 is 1-bit flipping
Good, but not good enough
We can do better

paul 🌲  ???
that's 2 locally
rhelmot 🌈 then there's a problem locally
paul 🌲 no, chad and I get the same thing
there's a problem remotely 😊
we're working on it
some register must be different
zwimer 🤡 Wait, I fucked up
rhelmot 🌈 got 2 remotely
subwire 🔥!!
nice!
Let's just fuzz offsets

paul 🍃 I'm fuzzing jump offsets in salls' 3 bit payload, should be able to get to 2

zwimer 🤡 We got 2
With lots of options
We got 1 !!!!

paul 🍃 HOLY SHIT P!!

@subwire

shortman 🕺 boom!!!!
BITS 64

_start:
  add ax, 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
### Final Scores

<table>
<thead>
<tr>
<th>TOTAL</th>
<th>Attack</th>
<th>Defense</th>
<th>KoH</th>
</tr>
</thead>
<tbody>
<tr>
<td>▲ 772 HITCON BFKinesiS</td>
<td>1006 HITCON BFKinesiS</td>
<td>159 A<em>0</em>E</td>
<td>664 Plaid Parliament of Pwning</td>
</tr>
<tr>
<td>▲ 590 Tea Deliverers</td>
<td>815 Tea Deliverers</td>
<td>156 HITCON BFKinesiS</td>
<td>477 A<em>0</em>E</td>
</tr>
<tr>
<td>▼ 564 A<em>0</em>E</td>
<td>656 m hack eroni</td>
<td>147 m hack eroni</td>
<td>459 Tea Deliverers</td>
</tr>
<tr>
<td>▲ 556 m hackeroni</td>
<td>646 Samurai</td>
<td>132 r3kapig</td>
<td>443 KaisHack GoN</td>
</tr>
<tr>
<td>▼ 399 Samurai</td>
<td>510 A<em>0</em>E</td>
<td>130 Tea Deliverers</td>
<td>405 Sauercloud</td>
</tr>
<tr>
<td>▼ 375 Sauercloud</td>
<td>499 r00timentary</td>
<td>127 Sauercloud</td>
<td>377 m hackeroni</td>
</tr>
<tr>
<td>▲ 359 r00timentary</td>
<td>339 SeoulPlusBadAss</td>
<td>111 r00timentary</td>
<td>370 SeoulPlusBadAss</td>
</tr>
<tr>
<td>▼ 331 SeoulPlusBadAss</td>
<td>292 saarsec</td>
<td>100 Samurai</td>
<td>333 TokyoWesterns</td>
</tr>
<tr>
<td>▼ 284 Shellphish</td>
<td>131 r3kapig</td>
<td>98 Shellphish</td>
<td>269 Shellphish</td>
</tr>
<tr>
<td>▼ 284 r3kapig</td>
<td>114 Sauercloud</td>
<td>88 KaisHack GoN</td>
<td>173 saarsec</td>
</tr>
<tr>
<td>▼ 281 KaisHack GoN</td>
<td>109 Shellphish</td>
<td>75 SeoulPlusBadAss</td>
<td>123 Samurai</td>
</tr>
<tr>
<td>▼ 235 saarsec</td>
<td>106 CGC</td>
<td>58 saarsec</td>
<td>59 CGC</td>
</tr>
<tr>
<td>▼ 215 TokyoWesterns</td>
<td>96 TokyoWesterns</td>
<td>54 TokyoWesterns</td>
<td>46 r00timentary</td>
</tr>
<tr>
<td>▲ 110 CGC</td>
<td>8 hxp</td>
<td>35 CGC</td>
<td>5 hxp</td>
</tr>
<tr>
<td>▼ 67 hxp</td>
<td>2 KaisHack GoN</td>
<td>34 hxp</td>
<td>0 r3kapig</td>
</tr>
</tbody>
</table>

\[
\Sigma_{\text{tick}} (1 \text{ for each stolen flag})
\]

\[
M_d = \max(\dagger, 100) = 1442
\]

\[
\Sigma_{\text{tick}} (1 \text{ if non-exploited AND there were exploits})
\]

\[
M_d = \max(\dagger, 100) = 213
\]

\[
M_k = \max(\dagger, 100) = 769
\]

\[
400^\gamma_M + 400^\delta_M + 200^\zeta_M
\]
10th Place!