Solutions

Cryptography

What is a Crypto?

- BerPGS{EBG13_Znlo3_J3_Fu0hyq_Hf3_Z0e3_E0haqf}
- OreCTF{ROT13_Mayb3_W3_Sh0uld_Us3_M0r3_R0unds}

This is a Crypto!

- T3JlQ1RGezBoX3MwX3RoNHRfMXNfYjRzMzY0fQ==
- OreCTF{0h_s0_th4t_1s_b4s364}

Bitwise Cryptography

- Use XOR with the file, first line is message, second is key
- OreCTF{X0R_1s_C00l}

On The Fence

- Given cipher decode it
- Title and description should hint you towards railfence cipher
- Brute force the key because it’s numbers, the actual key is 6
- OreCTF{The blaster bacons at midnight}

Hidden in Plain Sight

- based on the picoCTF challenge: Pixelated
- two images that need to be stacked onto each other to reveal the flag
- code can be written for this
  - me when chatgpt wrote the code for me <3
- flag:

oidua

- Idea: silly audio in Morse Code
- Solution:
  - download the .wav file which is in Morse Code
  - reverse the audio (should be obvious from the title)
  - put audio into a decoder for morse code or you could decode by hand??
  - ROT13 encoded!
    - Z0EF3P0QRVFFYYL
  - cyber chef that shawty for the flag
    - M0RS3C0DEISSLLY
  - wrap with OreCTF{}

Breaking The Code Book

- You’re given an aes encrypted file and a date range the file was encrypted with as well as knowing it’s the UNIX timestamp. Simply iterate over all possible timestamps and it’ll be there.
- openssl enc -d -aes-128-cbc -in fourth_wall_cipher -pass pass:$timestamp -nosalt
- Just change the $timestamp part in a loop and try to get the output
- OreCTF{Br3aking_c0d3s_1s_3asy}

Reverse Engineering

Director Ducky

- Ducky script makes .bin files that run on fake USB's that operate just like keyboards.
- Two options to solve this
  - DuckToolkit/ducktools.py at master · kevthehermit/DuckToolkit
    - Download this, and use the python code to decrypt it
    - python ducktools.py -d -l us location_of_bin answer.txt
- Use Ghidra
  - Set the language in Ghidra to be "ARV8"
  - Reverse Engineer the script to find the USB key codes
  - Convert the values for the correct language (us-qwerty)
  - Rebuild the Script
- STRING ORECTF{DUCK5_4r3_10UD}

Strings Author: Tristen

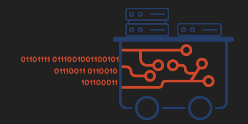
- Idea: String value set as global variable therefore it is stored in the data section of the binary.
- Solution: Run strings on the binary and scroll or grep for Ore.

```
[ublius@Ublius-Mini Downloads]$ strings Strings | grep Ore
OreCTF{S3tr1ngs_4r3_u53ful}
```

noStrings Author: Tristen

- Idea: String stored in a linked list that isn't saved to the data section.
- Solution: Load in Ghidra or Binary Ninja or any decompiler and see the char values assigned to every node.

```
char* rax = createNode('O')
*(rax + 8) = createNode('r')
void* rax_4 = *(rax + 8)
*(rax_4 + 8) = createNode('e')
void* rax_7 = *(rax_4 + 8)
*(rax_7 + 8) = createNode('C')
void* rax_10 = *(rax_7 + 8)
*(rax_10 + 8) = createNode('T')
void* rax_13 = *(rax_10 + 8)
*(rax_13 + 8) = createNode('F')
void* rax_16 = *(rax_13 + 8)
*(rax_16 + 8) = createNode('{')
void* rax_19 = *(rax_16 + 8)
*(rax_19 + 8) = createNode('N')
void* rax_22 = *(rax_19 + 8)
*(rax_22 + 8) = createNode('0')
void* rax_25 = *(rax_22 + 8)
*(rax_25 + 8) = createNode('_')
void* rax_28 = *(rax_25 + 8)
*(rax_28 + 8) = createNode('S')
void* rax_31 = *(rax_28 + 8)
*(rax_31 + 8) = createNode('3')
void* rax_34 = *(rax_31 + 8)
*(rax_34 + 8) = createNode('t')
void* rax_37 = *(rax_34 + 8)
*(rax_37 + 8) = createNode('r')
void* rax_40 = *(rax_37 + 8)
*(rax_40 + 8) = createNode('1')
void* rax_43 = *(rax_40 + 8)
*(rax_43 + 8) = createNode('n')
void* rax_46 = *(rax_43 + 8)
*(rax_46 + 8) = createNode('g')
void* rax_49 = *(rax_46 + 8)
*(rax_49 + 8) = createNode('s')
void* rax_52 = *(rax_49 + 8)
*(rax_52 + 8) = createNode('_')
void* rax_55 = *(rax_52 + 8)
*(rax_55 + 8) = createNode('4')
void* rax_58 = *(rax_55 + 8)
*(rax_58 + 8) = createNode('t')
void* rax_61 = *(rax_58 + 8)
*(rax_61 + 8) = createNode('t')
void* rax_64 = *(rax_61 + 8)
*(rax_64 + 8) = createNode('a')
void* rax_67 = *(rax_64 + 8)
*(rax_67 + 8) = createNode('c')
void* rax_70 = *(rax_67 + 8)
*(rax_70 + 8) = createNode('h')
void* rax_73 = *(rax_70 + 8)
*(rax_73 + 8) = createNode('e')
void* rax_76 = *(rax_73 + 8)
*(rax_76 + 8) = createNode('d')
void* rax_79 = *(rax_76 + 8)
*(rax_79 + 8) = createNode('}')
```



Password Protected Author: Nate

Idea:

you need a password to continue (or do you :O)

Solution:

Load into ghidra (or your fav dissasembler),

First look for the flag opening (I renamed flag_open):

look for a byte array with a print statement (I renamed byte_array):

Follow until end_of_flag variable, convert hex or decimal values to ascii

Wrap with OreCTF{ byte_array }

Finishes as: OreCTF{go is cool}

```
if (count == 679) {
  piStack_38 = &string___internal/abi.Type;
  success_string = &gostr_Good_job!,_here_is_the_flag:;
  file_descriptor.data = os.Stdout;
  file_descriptor.tab = &os::*os.File__implements__io.Writer___runtime.itab;
  a_01.len = 1;
  a_01.array = (interface_{} *)&piStack_38;
  a_01.cap = 1;
  fmt::fmt.Fprint(file_descriptor,a_01);
  piStack_48 = &string___internal/abi.Type;
  flag_open = &gostr_OreCTF{;
  w_00.data = os.Stdout;
  w_00.tab = &os::*os.File__implements__io.Writer___runtime.itab;
  a_02.len = 1;
  a_02.array = (interface_{} *)&piStack_48;
  a_02.cap = 1;
  fmt::fmt.Fprint(w_00,a_02);
  byte_array[0] = 0x67;
  byte_array[1] = 0x6f;
  byte_array[2] = 0x20;
  byte_array[3] = 0x69;
  byte_array[4] = 0x73;
  byte_array[5] = 0x20;
  byte_array[6] = 99;
  byte_array[7] = 0x6f;
  byte_array[8] = 0x6f;
  byte_array[9] = 0x6c;
  for (count = 0; count < 10; count = count + 1) {
    piStack_58 = piVar2;
    char_from_hex = (void *)uVar3;
    hex_val = runtime::runtime.intstring((uint8 (*) [4])0x0,(uint)byte_array[count]);
    char_from_hex = runtime::runtime.convTstring(hex_val);
    piStack_58 = &string___internal/abi.Type;
    w_02.data = os.Stdout;
    w_02.tab = &os::*os.File__implements__io.Writer___runtime.itab;
    a_04.len = 1;
    a_04.array = (interface_{} *)&piStack_58;
    a_04.cap = 1;
    fmt::fmt.Fprint(w_02,a_04);
  }
  piStack_68 = &string___internal/abi.Type;
  end_of_flag = &gostr_};
```

OSINT

Roarin’ Rochambeau Author: Bethany

Idea: Reverse search & logic

Solution: Use a reverse image search on the image, it will come up with a lot of info about the mine that used to be there. On the Wisconsin.gov page, one of the captions mentions that the mine is next to the Flambeau River, or you can find the name of the town (Ladysmith) and search up what river it’s on.

OreCTF{Flambeau_River}

Chasing The Storm Author: Garrett

- NyxVortex21 has a twitter account where a tweet can be found reference the bite of ‘87 from FNAF and him saying it’s his birthday year
- Answer is 1987

Photograph The Storm Author: Garrett

- The name is a reference to the fact that Nyx has an instagram account
- He made a comment on the oresec schedule for April saying he’s excited to go and that his favorite animal is a koala
- This can be found by nyxvortex21 on instagram -> follows -> oresec -> april schedule -> comments

Analyze The Storm Author: Garrett

- Nyx has a github account identified by 0xCrypt1c, you can google for this github account or search for it
- He has multiple different repos but all of the code is fake. You can tell after looking for a few seconds. So what is real then?
- Look at his readme commit history of his profile
- He’s interested in Monzy, which if you know you nerd core he has a famous song called kill dash 9!

I Will Live In Montana

- Title is a red october reference
- Find what port the fake submarine is at
- Go to what three words to find the three words
- Class of submarines around it can be identified via wikipedia or online sources
- OreCTF{KILO_SHUNTS_AVOIDING_DISCREPANCY}

Forensics

IoT Network Analysis 1-5 Author: Garrett

- Analyze using wireshark, just have to learn how to read IoT packets
- TpLink
- 10.0.0.11
- HS210
- 1.5.6
- 6
- Relatively easy once you learn how to read the packets in wireshark

Extract My Data

- Look at the exif data
- OreCTF{Golden}

Metaphysics

- Look at the metadata of the image
- OreCTF{M4k3_Sur3_t0_Str1p_M3t4d4t4}

Compress This Author: Garrett

- Run file and unzip repeatedly until you get the flag
- OreCTF{Z1p_B0mbs_4r3_F0r_Chumps}

Find Me Author: Garrett

- Download the APK file
- Unzip it because you can do that on APK’s
- You can then search through the entire thing for flags :)
- It’s under /res/color/flag.txt
- Decode the hex
- OreCTF{APKs_4r3_c00l}

Walking Author: Garrett

- Image of walking down the street with binary on the bottom, supposed to hint towards binwalk
- It’s just 2 jpg images concatenated together, you can extract the second image
- OreCTF{B1n_W4lk1ng_D0wn_Th3_Str33t}

Password Cracking

Pika Author: Garrett

- Simple MD5 password hash
- Crack it online or with rockyou
- megapikachu

Magic Man Author: Garrett

- SHA256 hash, probably can’t crack online but maybe can. It’s just a rockyou password
- abra kadabra alakazam

Water Gun! Author: Garrett

- Need to use john for this, identify the hash to be yescrypt and use rockyou
- mudkip

Cracking The Storm Author: Garrett

- All previous information about Nyx Vortex is required for this one
- You need monzy, koala, and 1987
- Write a program to combine monzy koala and 1987 in any order with underscores or without or with spaces or with all caps and each variation of that to get a wordlist
- You can then use the incisive leetspeak rule in hashcat to get the leetspeak for it
- m0nzy_k04l4_1987

MISC

IoT Basics 1-3 Author: Garrett

- Google + FCC ID
- Message Queuing Telemetry Transport
- Zigbee
- GN-1A-5LT

IIoT Basics 1-3 Author: Garrett

- Google
- Modbus
- Human Machine Interface
- Programmable Logic Controller

Imperial Army

- Font is Imperial from Madeon
- Find the font online somewhere, reverse image searching the entire image likely won’t work, so it’s best to split it into parts to try
- OreCTF{IMPERIAL}

Web Exploitation

Injectable Questionable

First link on the page, is not super useful but can be, Click the link and it says “Needs input”, use a get parameter with id, and it returns all the IDs!

You might notice a silly id, 1337.

You might also notice in the main page it says “You might need to ‘resolve’ something ;)”

Use the id you found (1337) in /resolve?1337

And badda bing, badda boom, there's your flag!