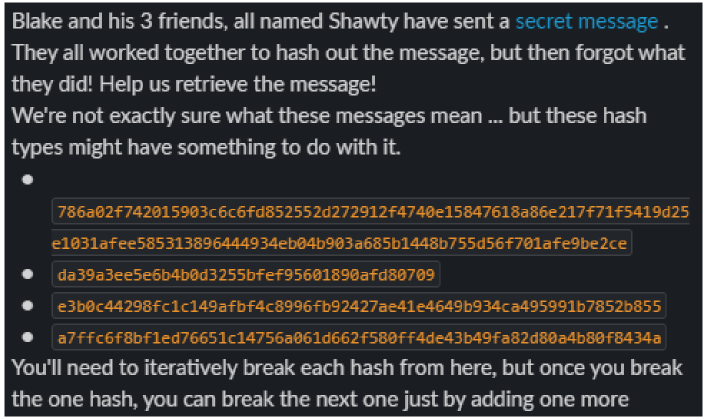
MetaCTF 2020/NCL FA20 Individual Game Debrief

Many Speakers!

MetaCTF CyberGames

Blake's Secret Message - Crypto 350

- Zaine
- "A @jtan20 Special"
- Iterative hashing, cycling through Blake2b, Sha1, Sha256, and Sha3-256



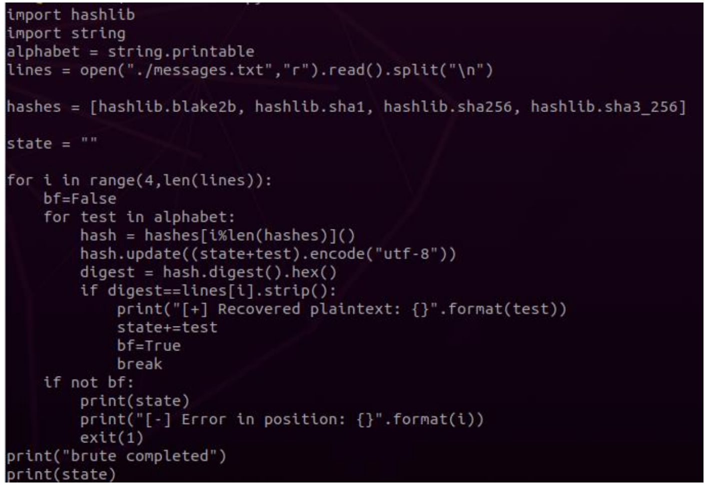
Blake and his 3 friends, all named Shawty have sent a secret message . They all worked together to hash out the message, but then forgot what they did! Help us retrieve the message!
We're not exactly sure what these messages mean ... but these hash types might have something to do with it.

- `786a02f742015903c6c6fd852552d272912f4740e15847618a86e217f71f5419d25e1031afee585313896444934eb04b903a685b1448b755d56f701afe9be2ce`
- `da39a3ee5e6b4b0d3255bfef95601890afd80709`
- `e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855`
- `a7ffc6f8bf1ed76651c14756a061d662f580ff4de43b49fa82d80a4b80f8434a`

You'll need to iteratively break each hash from here, but once you break the one hash, you can break the next one just by adding one more

Blake's Secret Message - Crypto 350

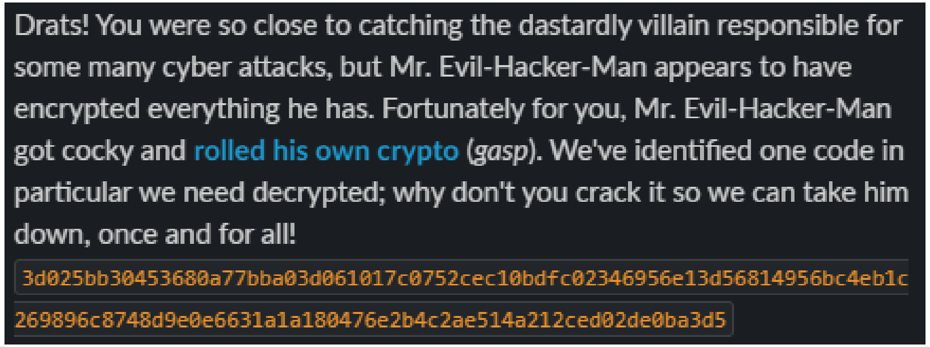
- First three hashes are the hashes of empty strings (these identified the algorithms in use for me)
- From there, the flag is hashed one character at a time, stepping between the three algorithms
- By brute forcing each character of the flag, we can reveal it byte by byte



```
import hashlib
import string
alphabet = string.printable
lines = open("./messages.txt","r").read().split("\n")

hashes = [hashlib.blake2b, hashlib.sha1, hashlib.sha256, hashlib.sha3_256]

state = ""

for i in range(4,len(lines)):
    bf=False
    for test in alphabet:
        hash = hashes[i%len(hashes)]()
        hash.update((state+test).encode("utf-8"))
        digest = hash.digest().hex()
        if digest==lines[i].strip():
            print("[+] Recovered plaintext: {}".format(test))
            state+=test
            bf=True
            break
    if not bf:
        print(state)
        print("[-] Error in position: {}".format(i))
        exit(1)
print("brute completed")
print(state)
```

A Block For Me, A Block For ??? - Crypto 400

- Zaine
- A Feistel Network with a relatively small keyspace, weak round function, and flawed padding
- Intended solve was to use the weak padding to perform a known-plaintext attack on a 32 bit keyspace
- Basically, just a brute force

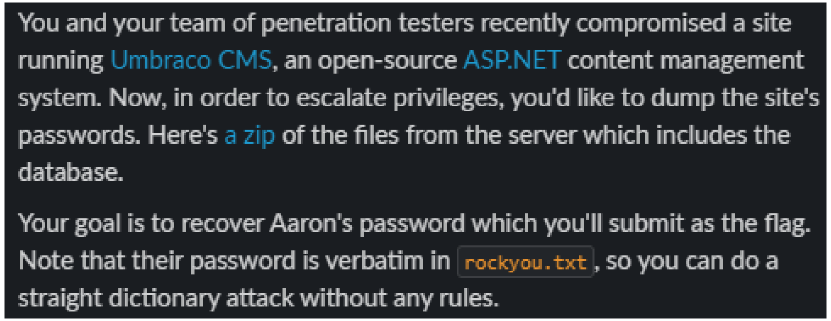


Drats! You were so close to catching the dastardly villain responsible for some many cyber attacks, but Mr. Evil-Hacker-Man appears to have encrypted everything he has. Fortunately for you, Mr. Evil-Hacker-Man got cocky and rolled his own crypto (*gasp*). We've identified one code in particular we need decrypted; why don't you crack it so we can take him down, once and for all!

```
3d025bb30453680a77bba03d061017c0752cec10bdfc02346956e13d56814956bc4eb1c
269896c8748d9e0e6631a1a180476e2b4c2ae514a212ced02de0ba3d5
```

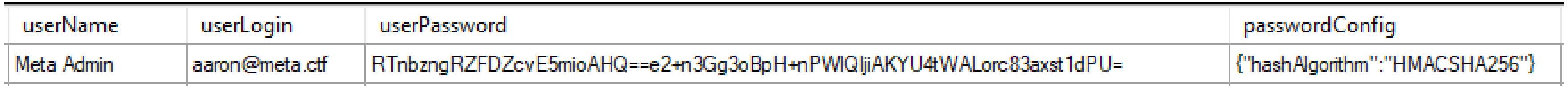
When Life Gives You Hashes - Crypto 525

- Zaine
- Finding the DB file, determining the hash format, and cracking the hash
- Download the Zip and search through it for any kind of database, find the .sdf database under App_Data
- Install an SDF viewer (SQL Compact Query Analyzer)
- Extract the hash and the algorithm used



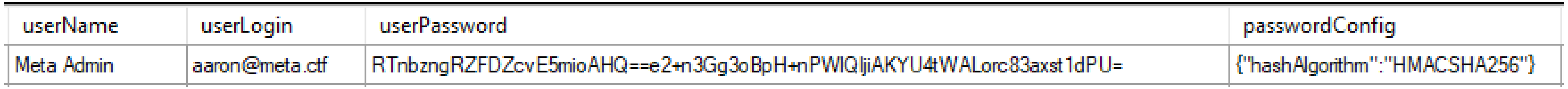
You and your team of penetration testers recently compromised a site running Umbraco CMS, an open-source ASP.NET content management system. Now, in order to escalate privileges, you'd like to dump the site's passwords. Here's a zip of the files from the server which includes the database.

Your goal is to recover Aaron's password which you'll submit as the flag. Note that their password is verbatim in `rockyou.txt`, so you can do a straight dictionary attack without any rules.



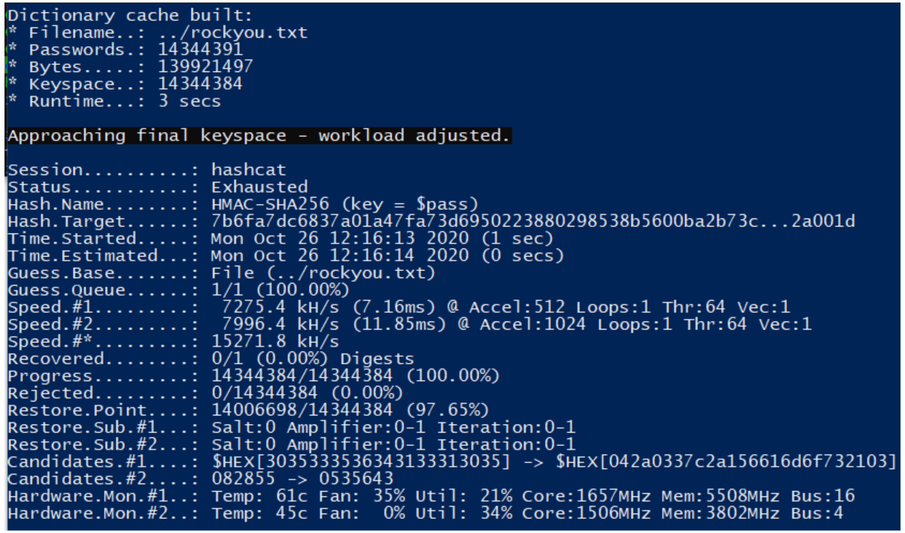
userName	userLogin	userPassword	passwordConfig
Meta Admin	aaron@meta.ctf	RTnbzngRZFDZcvE5mioAHQ==e2+n3Gg3oBpH+nPWIQljiAKYU4tWALorc83axst1dPU=	{"hashAlgorithm":"HMACSHA256"}

When Life Gives You Hashes - Crypto 525



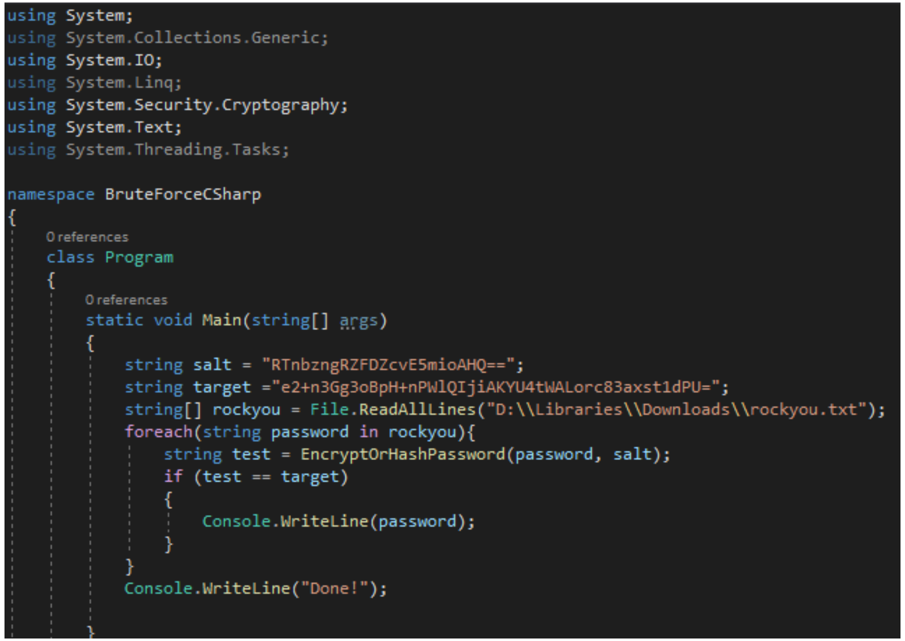
userName	userLogin	userPassword	passwordConfig
Meta Admin	aaron@meta.ctf	RTnbzngRZFDZcvE5mioAHQ==e2+n3Gg3oBpH+nPWIQljiAKYU4tWALorc83axst1dPU=	{"hashAlgorithm":"HMACSHA256"}

- Idea 1: Use either hashcat or JTR to crack the HMAC, retrieving the key
- Result: Sadness
- Idea 2: scrape the hashing code out of the CMS's git repo and use it in a custom cracker
- Result: Success!

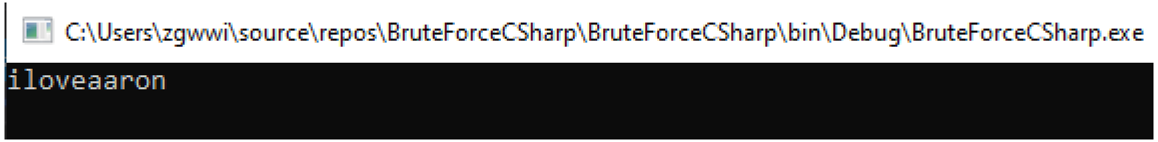


```
Dictionary cache built:
* Filename..: ../rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 3 secs

Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Name........: HMAC-SHA256 (key = $pass)
Hash.Target......: 7b6fa7dc6837a01a47fa73d6950223880298538b5600ba2b73c...2a001d
Time.Started.....: Mon Oct 26 12:16:13 2020 (1 sec)
Time.Estimated...: Mon Oct 26 12:16:14 2020 (0 secs)
Guess.Base.......: File (../rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  7275.4 kH/s (7.16ms) @ Accel:512 Loops:1 Thr:64 Vec:1
Speed.#2.........:  7996.4 kH/s (11.85ms) @ Accel:1024 Loops:1 Thr:64 Vec:1
Speed.#*.........: 15271.8 kH/s
Recovered........: 0/1 (0.00%) Digests
Progress.........: 14344384/14344384 (100.00%)
Rejected.........: 0/14344384 (0.00%)
Restore.Point....: 14006698/14344384 (97.65%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: $HEX[3035333536343133313035] -> $HEX[042a0337c2a156616d6f732103]
Candidates.#2....: 082855 -> 0535643
Hardware.Mon.#1..: Temp: 61c Fan: 35% Util: 21% Core:1657MHz Mem:5508MHz Bus:16
Hardware.Mon.#2..: Temp: 45c Fan:  0% Util: 34% Core:1506MHz Mem:3802MHz Bus:4
```

When Life Gives You Hashes - Crypto 525



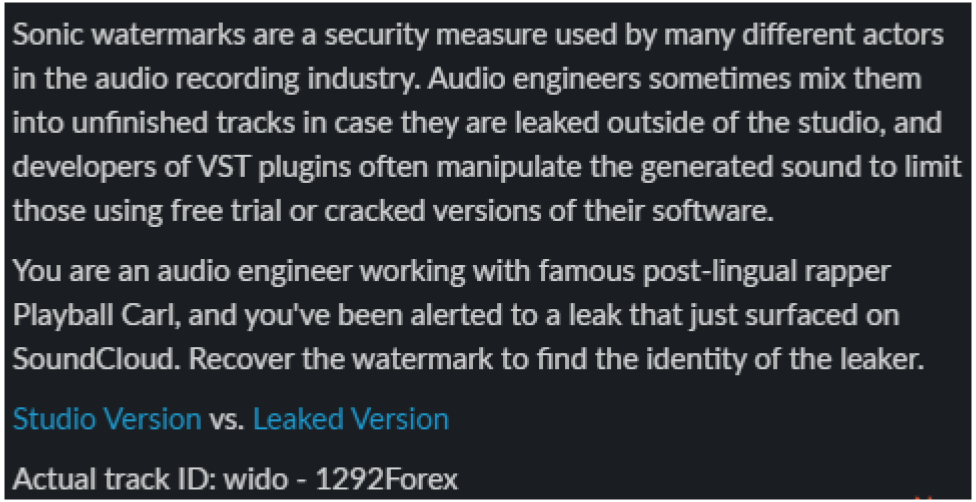
```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace BruteForceCSharp
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            string salt = "RTnbzngRZFDZcvE5mioAHQ==";
            string target ="e2+n3Gg3oBpH+nPWlQIjiAKYU4tWALorc83axst1dPU=";
            string[] rockyou = File.ReadAllLines("D:\\Libraries\\Downloads\\rockyou.txt");
            foreach(string password in rockyou){
                string test = EncryptOrHashPassword(password, salt);
                if (test == target)
                {
                    Console.WriteLine(password);
                }
            }
            Console.WriteLine("Done!");

        }
```



```
C:\Users\zgwwi\source\repos\BruteForceCSharp\BruteForceCSharp\bin\Debug\BruteForceCSharp.exe
iloveaaron
```

Watermarked - Other 250

- Zaine
- Recovering a sonic watermark
- Given two audio tracks
- They can be subtracted from each other using Audacity, revealing any differences (which reveals the watermark)



Sonic watermarks are a security measure used by many different actors in the audio recording industry. Audio engineers sometimes mix them into unfinished tracks in case they are leaked outside of the studio, and developers of VST plugins often manipulate the generated sound to limit those using free trial or cracked versions of their software.

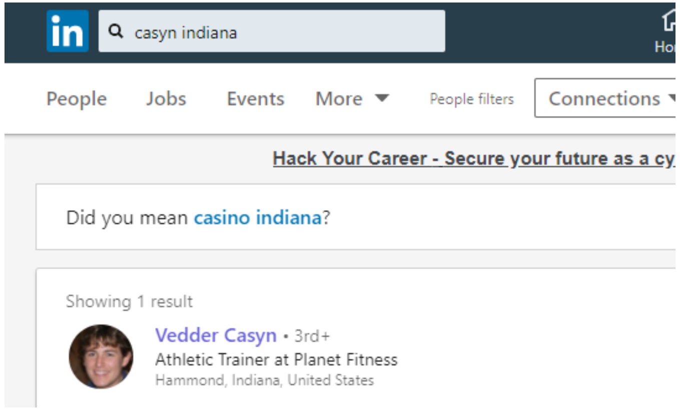
You are an audio engineer working with famous post-lingual rapper Playball Carl, and you've been alerted to a leak that just surfaced on SoundCloud. Recover the watermark to find the identity of the leaker.

Studio Version vs. Leaked Version

Actual track ID: wido - 1292Forex

Finding Mr. Casyn - Recon 275

- @mod
- We're looking for a Mr. Casyn, who has been reported missing. We believe he lives in the Chicagoland area, but don't think he's in Illinois proper. We need your help finding him and identifying the right Mr. Casyn will help us begin our search.
- The flag for this challenge is the first name of Mr. Casyn

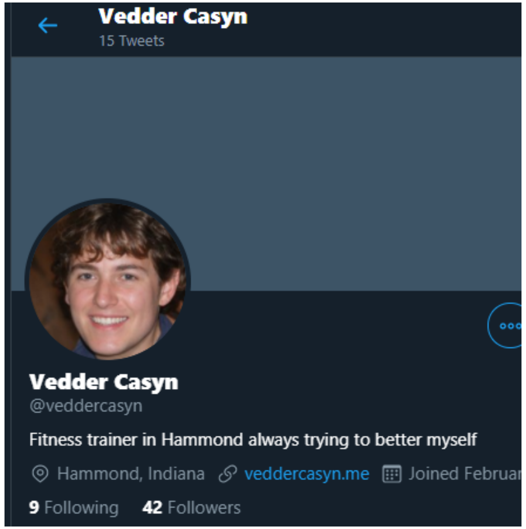


Hack Your Career - Secure your future as a cy

Did you mean casino indiana?

Showing 1 result

Vedder Casyn • 3rd+
Athletic Trainer at Planet Fitness
Hammond, Indiana, United States



Vedder Casyn
15 Tweets

Vedder Casyn
@veddercasyn

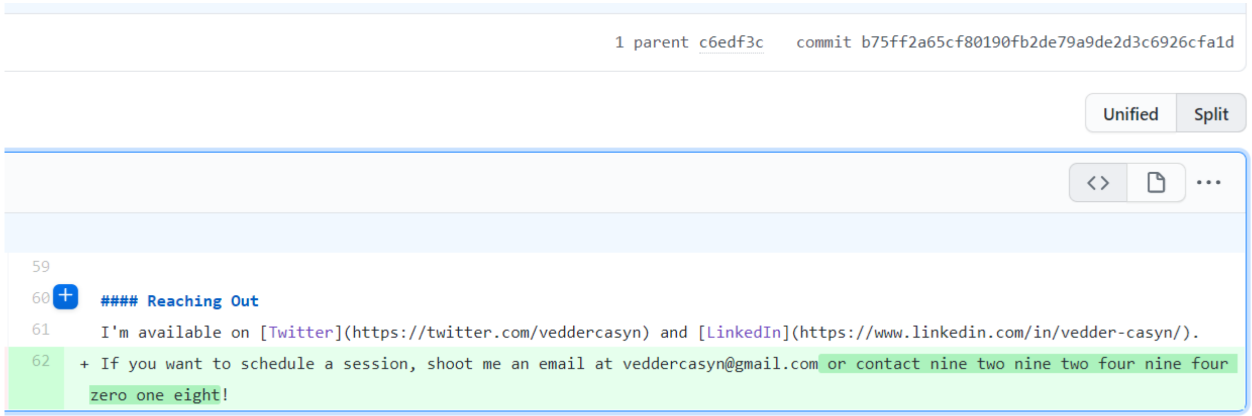
Fitness trainer in Hammond always trying to better myself

Hammond, Indiana veddercasyn.me Joined Februar

9 Following 42 Followers

Ring Ring - Recon 325

- @mod
- We want to try and reach out to Mr. Casyn via telephone. Can you figure out his phone number?
- Flag format: XXX-XXX-XXXX.
- LinkedIn and Twitter have a link to Website which has a github link
- Check history on github

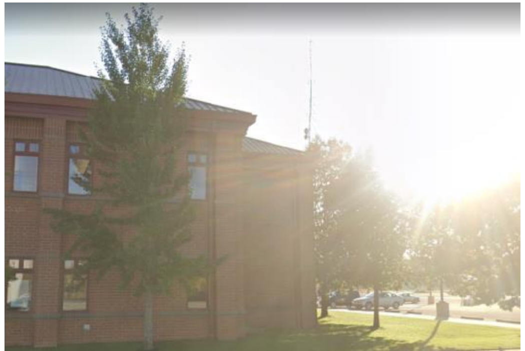


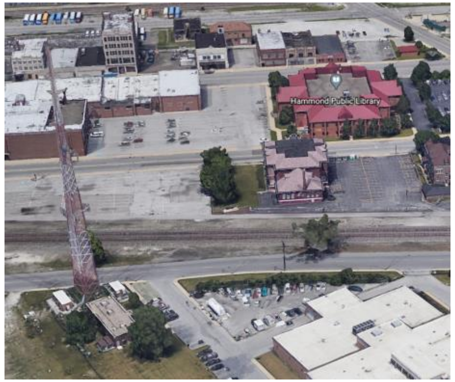
```
1 parent c6edf3c    commit b75ff2a65cf80190fb2de79a9de2d3c6926cfa1d
```

```
  #### Reaching Out
  I'm available on [Twitter](https://twitter.com/veddercasyn) and [LinkedIn](https://www.linkedin.com/in/vedder-casyn/).
+ If you want to schedule a session, shoot me an email at veddercasyn@gmail.com or contact nine two nine two four nine four
  zero one eight!
```

Hangout Spots - Recon 525

- @mod
- Can you find out where he likes to frequently hang out so we can look for clues of where he's been recently? Once you find the image, think of how we can use what we know to geolocate the image based on what's in the picture.
- Image is on git change history
- Identify where the radio tower in the image is, I spent a long time looking at google maps at spot where FCC had registered towers
- Contact people in the town that will respond, thought to check with a librarian, turns it was at the library/Use google earth to view the town and find tower





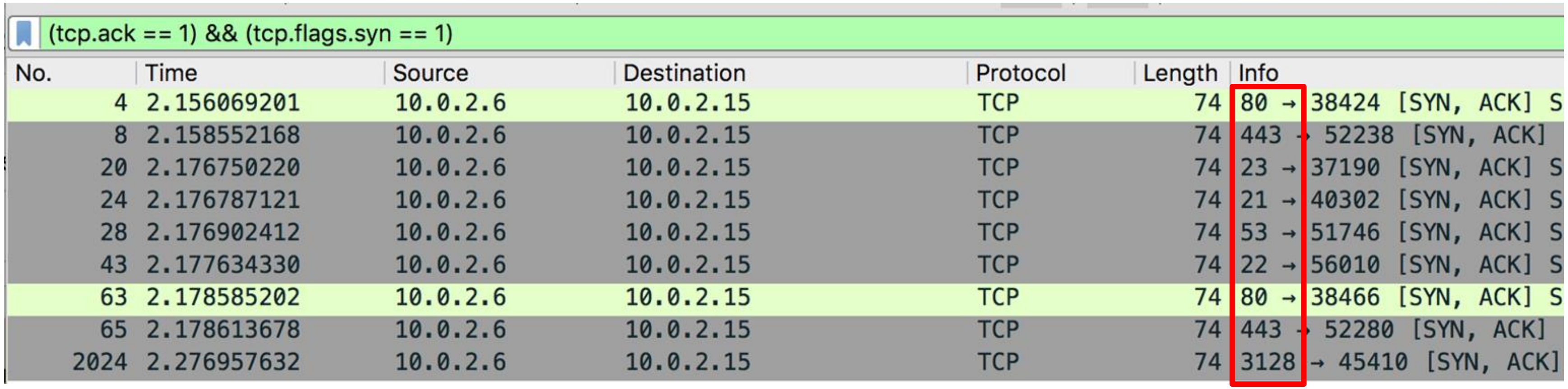
Open Thermal Exhaust Port - Forensics 275

- @cyu

Prompt: Our TCP connect Nmap scan found some open ports it seems. We may only have a pcap of the traffic, but I'm sure that won't be a problem! Can you tell us which ones they are? The flag will be the sum of the open ports. For example, if ports 25 and 110 were open, the answer would be MetaCTF{135}.

Solution: The pcap is a large Nmap scan. You need to know how Nmap TCP scans work. If the port is open, there will be a SYN-ACK (acceptance), otherwise, there will just be a RST (rejection). Thus, we can solve this with a simple Wireshark filter of **(tcp.ack == 1) && (tcp.flags.syn == 1)**. This will capture the second packet in a successful three-way handshake.

Open Thermal Exhaust Port - Forensics 275



(tcp.ack == 1) && (tcp.flags.syn == 1)

No.	Time	Source	Destination	Protocol	Length	Info
4	2.156069201	10.0.2.6	10.0.2.15	TCP	74	80 → 38424 [SYN, ACK] S
8	2.158552168	10.0.2.6	10.0.2.15	TCP	74	443 → 52238 [SYN, ACK]
20	2.176750220	10.0.2.6	10.0.2.15	TCP	74	23 → 37190 [SYN, ACK] S
24	2.176787121	10.0.2.6	10.0.2.15	TCP	74	21 → 40302 [SYN, ACK] S
28	2.176902412	10.0.2.6	10.0.2.15	TCP	74	53 → 51746 [SYN, ACK] S
43	2.177634330	10.0.2.6	10.0.2.15	TCP	74	22 → 56010 [SYN, ACK] S
63	2.178585202	10.0.2.6	10.0.2.15	TCP	74	80 → 38466 [SYN, ACK] S
65	2.178613678	10.0.2.6	10.0.2.15	TCP	74	443 → 52280 [SYN, ACK]
2024	2.276957632	10.0.2.6	10.0.2.15	TCP	74	3128 → 45410 [SYN, ACK]

The ports that connected successfully are 80, 443, 23, 21, 53, 22, and 3128.

Add up all the ports = 3770

Flag: MetaCTF{3770}

Mystery C2 - Forensics 325

- @cyu, have @templar talk about this if @cyu is not around

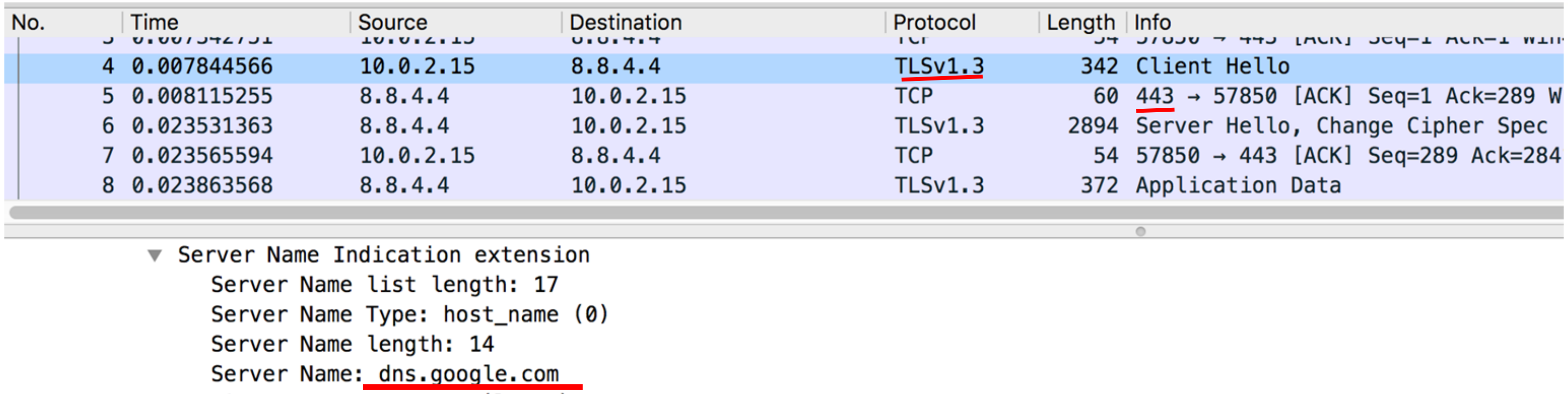
Prompt: Our threat intel team detected some malicious Command-and-Control traffic in our network. Can you identify what C2 framework the threat actor is using?

Solution: This is more Pcap analysis very similar to “C2 Channels” on TCTF.

First, the traffic is clearly encrypted over HTTPS, but a lot of C2s do that, so that isn’t super helpful.

If we look at the IPs, the traffic is going to 8.8.4.4...which is google’s DNS server

Mystery C2 - Forensics 325



```
No.  Time          Source      Destination  Protocol  Length  Info
  4  0.007844566   10.0.2.15   8.8.4.4      TLSv1.3      342  Client Hello
  5  0.008115255   8.8.4.4     10.0.2.15    TCP           60  443 → 57850 [ACK] Seq=1 Ack=289 W
  6  0.023531363   8.8.4.4     10.0.2.15    TLSv1.3     2894  Server Hello, Change Cipher Spec
  7  0.023565594   10.0.2.15   8.8.4.4      TCP           54  57850 → 443 [ACK] Seq=289 Ack=284
  8  0.023863568   8.8.4.4     10.0.2.15    TLSv1.3      372  Application Data
```

```
▼ Server Name Indication extension
    Server Name list length: 17
    Server Name Type: host_name (0)
    Server Name length: 14
    Server Name: dns.google.com
```

So there's TLS encrypted DNS traffic going over port 443. Sounds like DNS-over-HTTPS to me! Let's see if there are any C2s that use DNS over HTTPS as a communication protocol.

Googling for DNS over HTTPS C2 results in one option: GoDoH.

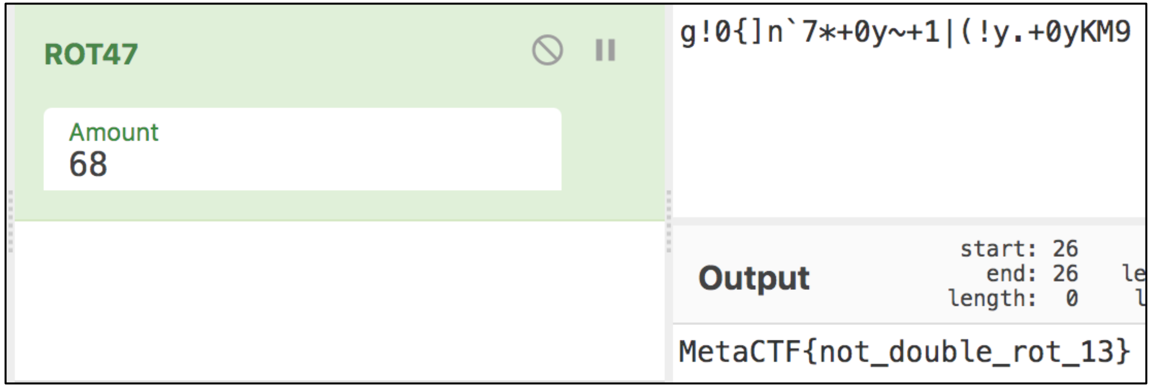
Alternatively, this can be solved with JA3 because GoDoH uses a very specific Go requests library to do TLS/SSL.

ROT 26 - Cryptography 150

- @cyu

ROT26 was applied to encrypt this ciphertext: g!0{]n`7*+0y~+1|(!y.+0yKM9

Solution: CyberChef ROT 47 - has a keyspace of 94. If ROT 26 was used to encrypt, 94-26=68 must be used to decrypt.



ROT47

Amount
68

```
g!0{]n`7*+0y~+1|(!y.+0yKM9
```

Output

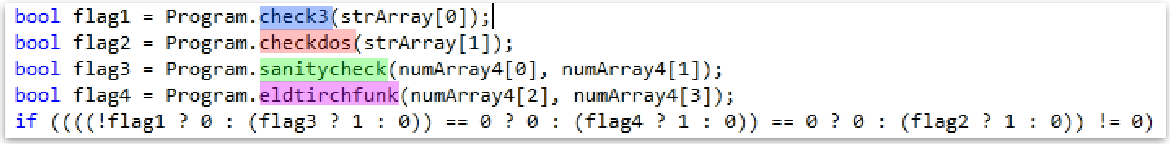
```
 start: 26
   end: 26
length:  0
```

```
MetaCTF{not_double_rot_13}
```

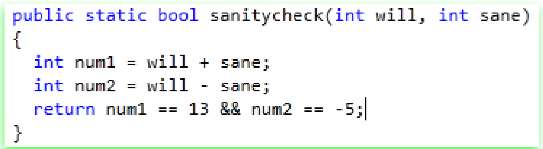
NCL Fall 2020 Individual Game

Dotinator - Enum/Exploit 3

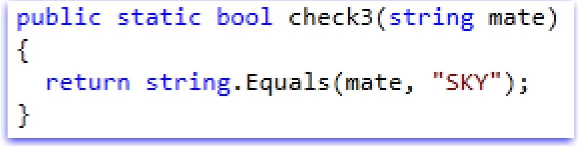
@Robert Weiner



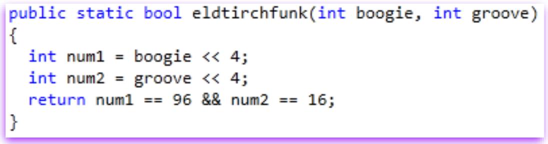
```
bool flag1 = Program.check3(strArray[0]);
bool flag2 = Program.checkdos(strArray[1]);
bool flag3 = Program.sanitycheck(numArray4[0], numArray4[1]);
bool flag4 = Program.eldtirchfunk(numArray4[2], numArray4[3]);
if ((((!flag1 ? 0 : (flag3 ? 1 : 0)) == 0 ? 0 : (flag4 ? 1 : 0)) == 0 ? 0 : (flag2 ? 1 : 0)) != 0)
```



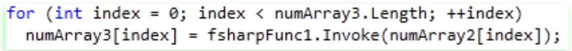
```
public static bool sanitycheck(int will, int sane)
{
  int num1 = will + sane;
  int num2 = will - sane;
  return num1 == 13 && num2 == -5;
}
```



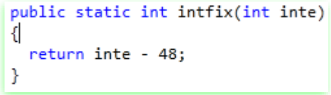
```
public static bool check3(string mate)
{
  return string.Equals(mate, "SKY");
}
```



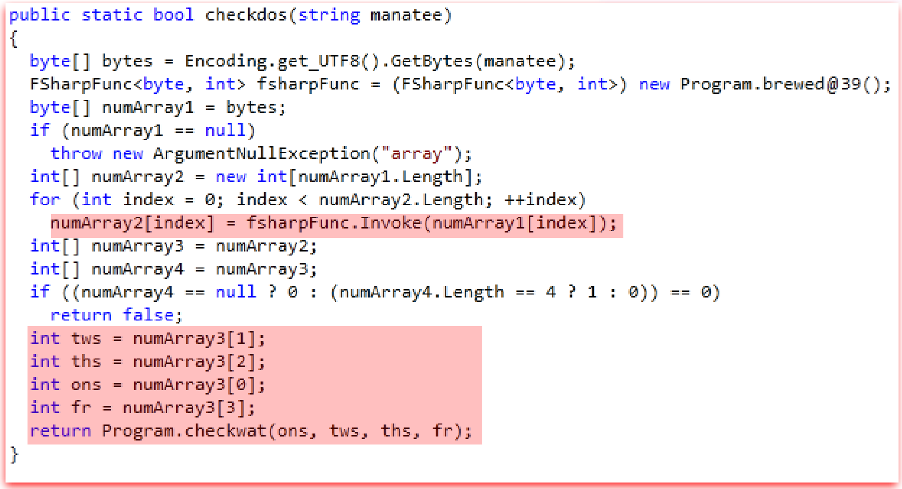
```
public static bool eldtirchfunk(int boogie, int groove)
{
  int num1 = boogie << 4;
  int num2 = groove << 4;
  return num1 == 96 && num2 == 16;
}
```



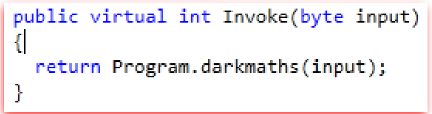
```
for (int index = 0; index < numArray3.Length; ++index)
  numArray3[index] = fsharpFunc1.Invoke(numArray2[index]);
```



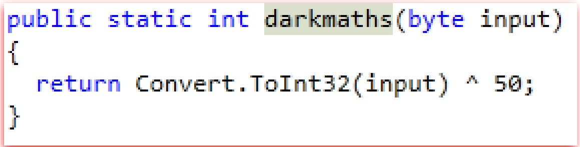
```
public static int intfix(int inte)
{
  return inte - 48;
}
```



```
public static bool checkdos(string manatee)
{
  byte[] bytes = Encoding.get_UTF8().GetBytes(manatee);
  FSharpFunc<byte, int> fsharpFunc = (FSharpFunc<byte, int>) new Program.brewed@39();
  byte[] numArray1 = bytes;
  if (numArray1 == null)
    throw new ArgumentNullException("array");
  int[] numArray2 = new int[numArray1.Length];
  for (int index = 0; index < numArray2.Length; ++index)
    numArray2[index] = fsharpFunc.Invoke(numArray1[index]);
  int[] numArray3 = numArray2;
  int[] numArray4 = numArray3;
  if ((numArray4 == null ? 0 : (numArray4.Length == 4 ? 1 : 0)) == 0)
    return false;
  int tws = numArray3[1];
  int ths = numArray3[2];
  int ons = numArray3[0];
  int fr = numArray3[3];
  return Program.checkwat(ons, tws, ths, fr);
}
```

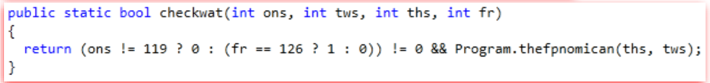


```
public virtual int Invoke(byte input)
{
  return Program.darkmaths(input);
}
```

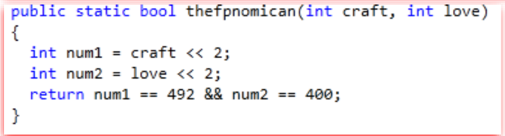


```
public static int darkmaths(byte input)
{
  return Convert.ToInt32(input) ^ 50;
}
```

This doesn't even show all the transformations and checks on the input



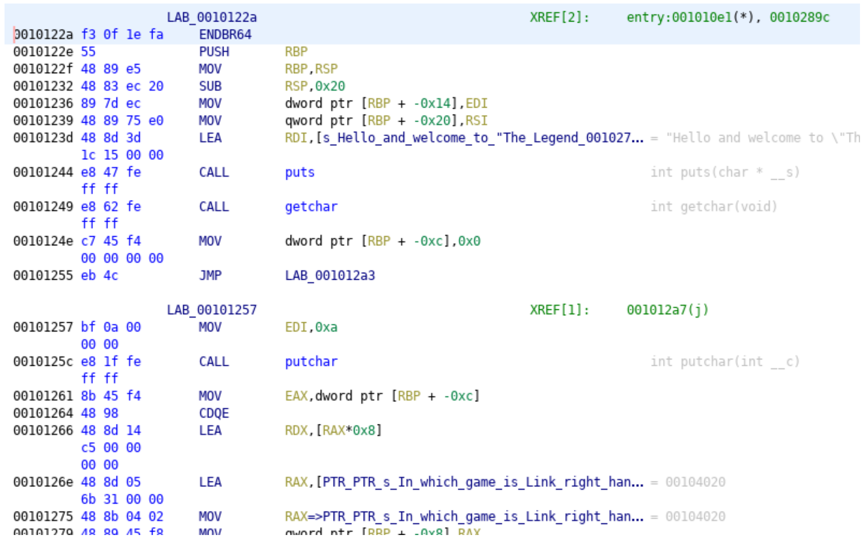
```
public static bool checkwat(int ons, int tws, int ths, int fr)
{
  return (ons != 119 ? 0 : (fr == 126 ? 1 : 0)) != 0 && Program.thefpnomican(ths, tws);
}
```



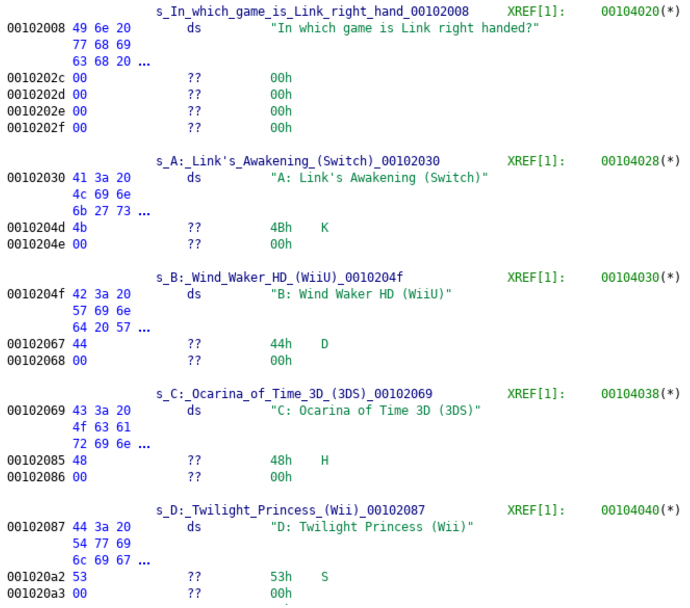
```
public static bool thefpnomican(int craft, int love)
{
  int num1 = craft << 2;
  int num2 = love << 2;
  return num1 == 492 && num2 == 400;
}
```

Trivia Game - Enum/Exploit 5

@Robert Weiner



```
                             LAB_0010122a                                    XREF[2]:     entry:001010e1(*), 0010289c
0010122a f3 0f 1e fa         ENDBR64
0010122e 55                  PUSH       RBP
0010122f 48 89 e5            MOV        RBP,RSP
00101232 48 83 ec 20         SUB        RSP,0x20
00101236 89 7d ec            MOV        dword ptr [RBP + -0x14],EDI
00101239 48 89 75 e0         MOV        qword ptr [RBP + -0x20],RSI
0010123d 48 8d 3d            LEA        RDI,[s_Hello_and_welcome_to_"The_Legend_001027...  = "Hello and welcome to \"Th
         1c 15 00 00
00101244 e8 47 fe            CALL       puts                                        int puts(char * __s)
         ff ff
00101249 e8 62 fe            CALL       getchar                                     int getchar(void)
         ff ff
0010124e c7 45 f4            MOV        dword ptr [RBP + -0xc],0x0
         00 00 00 00
00101255 eb 4c               JMP        LAB_001012a3

                             LAB_00101257                                    XREF[1]:     001012a7(j)
00101257 bf 0a 00            MOV        EDI,0xa
         00 00
0010125c e8 1f fe            CALL       putchar                                     int putchar(int __c)
         ff ff
00101261 8b 45 f4            MOV        EAX,dword ptr [RBP + -0xc]
00101264 48 98               CDQE
00101266 48 8d 14            LEA        RDX,[RAX*0x8]
         c5 00 00
         00 00
0010126e 48 8d 05            LEA        RAX,[PTR_PTR_s_In_which_game_is_Link_right_han...  = 00104020
         6b 31 00 00
00101275 48 8b 04 02         MOV        RAX=>PTR_PTR_s_In_which_game_is_Link_right_han...  = 00104020
00101279 48 89 45 f8         MOV        qword ptr [RBP + -0x8],RAX
```



```
                             s_In_which_game_is_Link_right_hand_00102008      XREF[1]:     00104020(*)
00102008 49 6e 20            ds          "In which game is Link right handed?"
         77 68 69
         63 68 20 ...
0010202c 00                  ??          00h
0010202d 00                  ??          00h
0010202e 00                  ??          00h
0010202f 00                  ??          00h

                             s_A:_Link's_Awakening_(Switch)_00102030          XREF[1]:     00104028(*)
00102030 41 3a 20            ds          "A: Link's Awakening (Switch)"
         4c 69 6e
         6b 27 73 ...
0010204d 4b                  ??          4Bh    K
0010204e 00                  ??          00h

                             s_B:_Wind_Waker_HD_(WiiU)_0010204f               XREF[1]:     00104030(*)
0010204f 42 3a 20            ds          "B: Wind Waker HD (WiiU)"
         57 69 6e
         64 20 57 ...
00102067 44                  ??          44h    D
00102068 00                  ??          00h

                             s_C:_Ocarina_of_Time_3D_(3DS)_00102069           XREF[1]:     00104038(*)
00102069 43 3a 20            ds          "C: Ocarina of Time 3D (3DS)"
         4f 63 61
         72 69 6e ...
00102085 48                  ??          48h    H
00102086 00                  ??          00h

                             s_D:_Twilight_Princess_(Wii)_00102087            XREF[1]:     00104040(*)
00102087 44 3a 20            ds          "D: Twilight Princess (Wii)"
         54 77 69
         6c 69 67 ...
001020a2 53                  ??          53h    S
001020a3 00                  ??          00h
```

Doesn't actually call any main function, just jumps into instruction, so no decompilation

Flag is made up of single chars mapped to the correct answers to the trivia questions stored in the .data section

Proud Sponsors

CACI

EVER VIGILANT

VOLGENAU SCHOOL

MASON

ENGINEERING