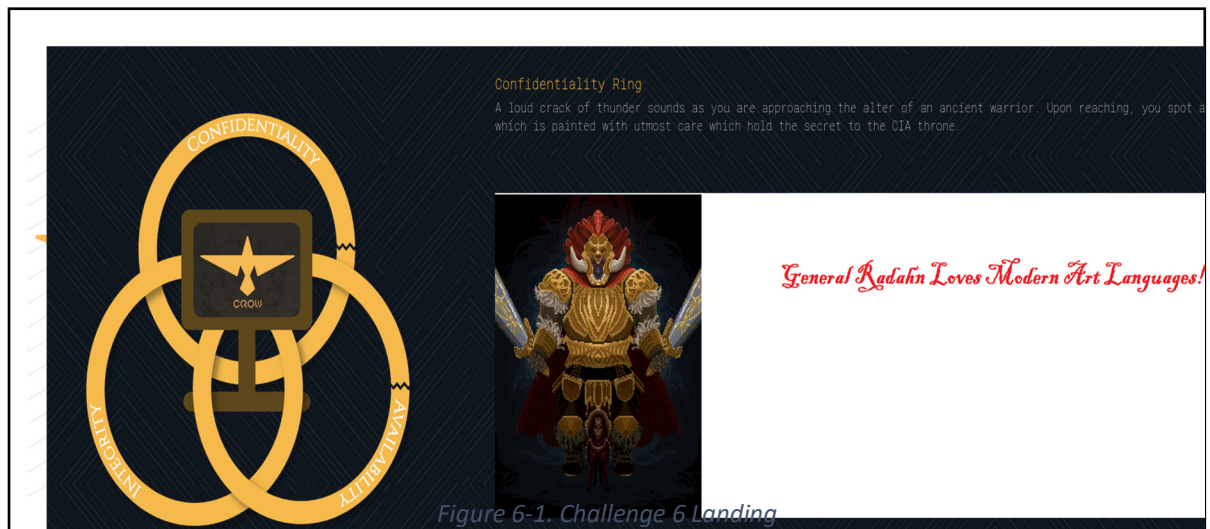


Challenge 6

Name of the Challenge	Magical Art
Type of Challenge	Piet (Esoteric Programming Languages)
Difficulty Level	Medium
Tool/Reference	Piet Decompiler



Solution:

1. Save the image from the landing page to your machine.



General Radahn Loves Modern Art Languages!

Figure 6-2. Challenge Image.

2. Take a closer look at the challenge image from the starting pixel. You can find a colour pattern on the top.

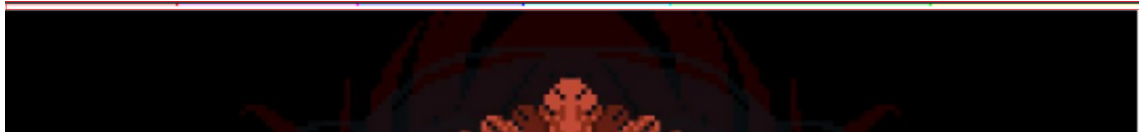


Figure 6-3. A Closer Look.

3. Upon further research on the internet. You will find that the challenge can be solved using a piet decompiler.


← → ↺ 🏠 <https://www.bertnase.de/npiet/npiet-execute.php>

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Hi,
welcome to [npiet online](#) !
Please upload a [piet program](#) image and [npiet](#) will execute it and display the result.

Hello world! (click me) Prime number test (click me)

Please give it a try - and have fun !

1. Choose a File:	<input type="button" value="Browse..."/> No file selected.
1b. Check the captcha:	<input type="checkbox"/> I'm not a robot  <small>reCAPTCHA Privacy - Terms</small>
2a. Launch immediately:	<input type="button" value="Upload and execute"/>
2b. Launch, but then let me add some input to pass !	<input type="button" value="Upload and ask about input"/>

back to [npiet online](#) - try again !
back to [npiet](#)
back to [bertnase.de](#)

And sorry about the captcha, but there were too many files with select and union in the name...

Figure 6-4. Piet Decompiler.

4. The challenge image is uploaded, and the interpreter gives out the following result:

Info: executing: npiet -w -e 220000 6_2.png

RUNE{h51rcf6o8ynu7zp9}

Figure 6-4. Final Flag.

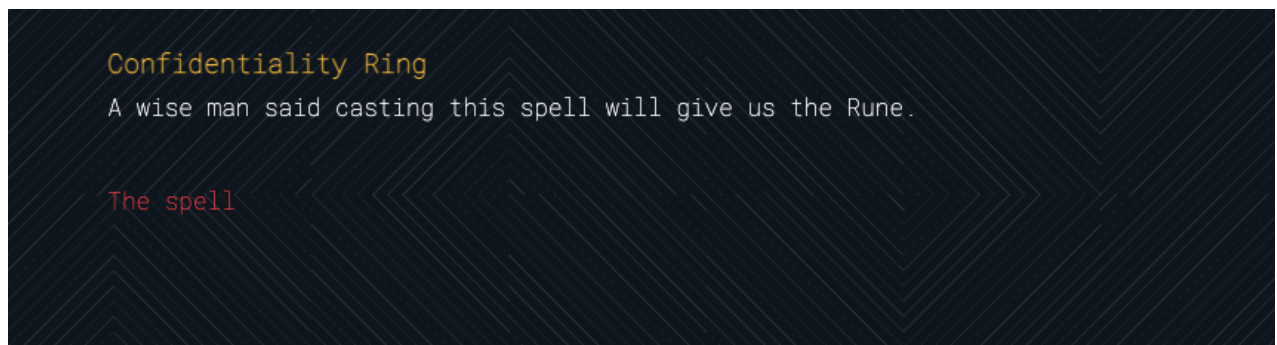
FLAG: RUNE{ h51rcf6o8ynu7zp9}

Challenge 14

Name of the Challenge	Strings and XOR
Type of Challenge	Strings and XOR
Difficulty Level	Easy
Tool/Reference	Cyber Chef

Problem:

The participant is provided with a link to an executable file `getRune.out`



Solution:

1. Use *strings* to find the literal strings in the executable

```
vkumar@sparrow Desktop % strings getRune.out
gD1001GliaK
ffN6bqrst12
51_UKps]X%xWT~w'57BE`0
I have the flag
Do not sway but the fear is abound
and the fell voice of Mardon said
Zero with Zero is Zero and One with One is Zero
yet Zero with One is One and One with Zero is One still
Thisisnotthekey
ButthereisAkey
```

2. You will notice that the first two random strings are 11 characters in size and the third one is 22 characters
3. It gives you a clue that the first two strings can be concatenated together and used

in a characterwise/bitwise operation with the third one

4. Concatenate the first two strings and XOR with the third one gives you the output
Rune{A411D3120AEDE11Q}

Challenge 16

Name of the Challenge	Holy Trinity
Type of Challenge	BrainFuck + GeoJson
Difficulty Level	Hard
Tool/Reference	BrainFuck Decompiler, QGIS



Solution:

1. The challenge file is initially analysed using the command “file”, which helps determine the type of a file and its data. The result of this command shows that the file is ASCII text which can be read in the terminal.

```
(kali㉿kali)-[~/chal_create/Writeups/R0/C16]
$ file Wizards_Wisdom
Wizards_Wisdom: ASCII text, with very long lines (65536), with no line terminators

(kali㉿kali)-[~/chal_create/Writeups/R0/C16]
$ cat Wizards_Wisdom
--[→++++<]>.>+++++++.[→+++<]>+.[→+++<]>+++++.-----.[→+++<]>+
.+++++.---[→+++<]>.>+++++++.[→+++<]>+.[→+++<]>+
.[→+++<]>+++++.---.+++++.-----.[→+++<]>+.[→+++<]>+
- [→+++<]>.>+++++++.[→+++<]>+.[→+++<]>+.[→+++<]>+++++.-----.[→+++<]>+
+++[→+++<]>+.
```

Figure 16-2. Analysis of the challenge file.

- The text given in the challenge is of BrainFuck programming language and can be interpreted using a [BrainFuck Interpreter](#). Upon Interpreting, we see that text is in GeoJson file format.

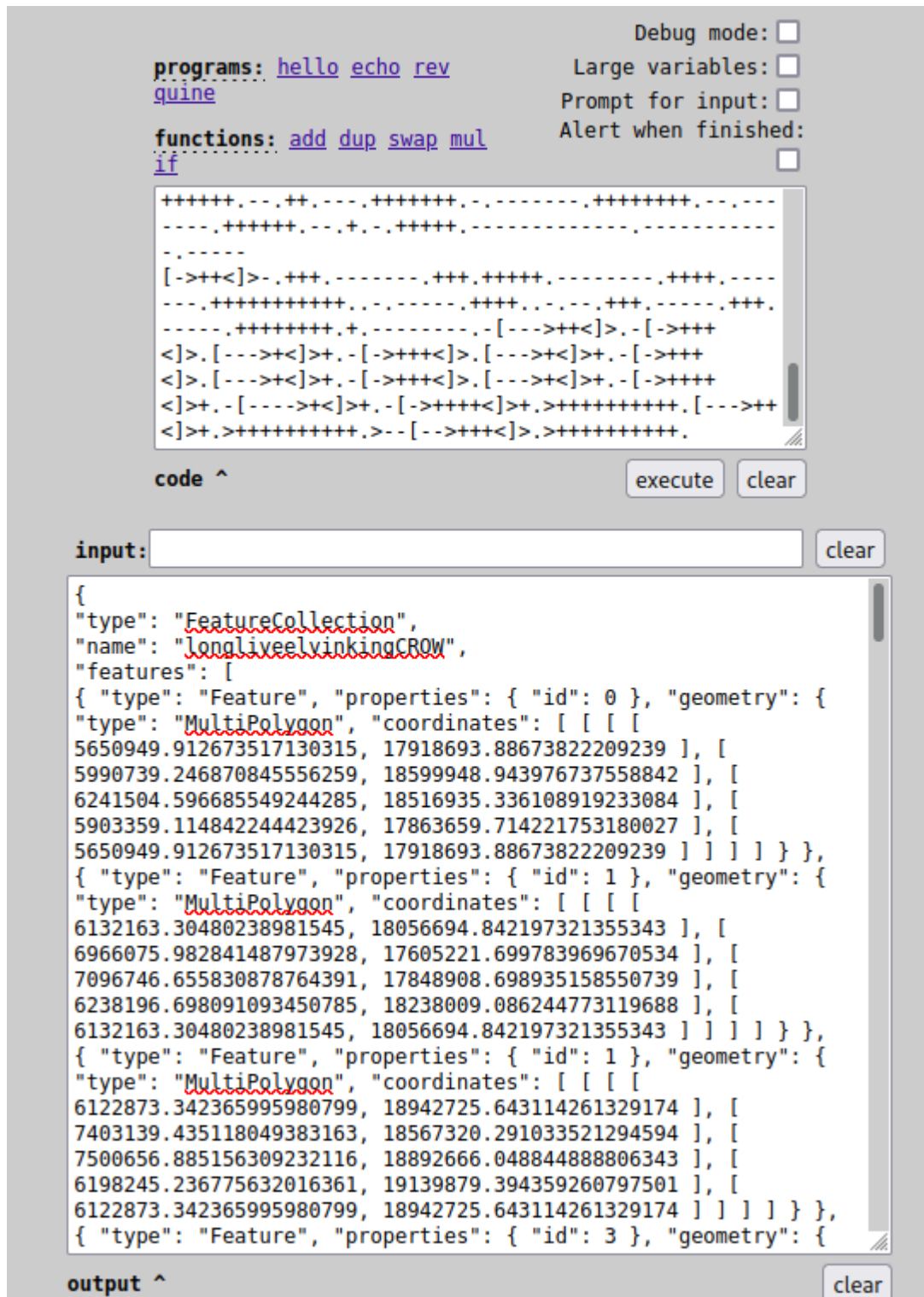


Figure 16-3. BrainFuck Interpreter.

3. The GeoJson data can be visualised in the open-source software QGIS. Upon launching the QGIS software, Import the GeoJson challenge data.

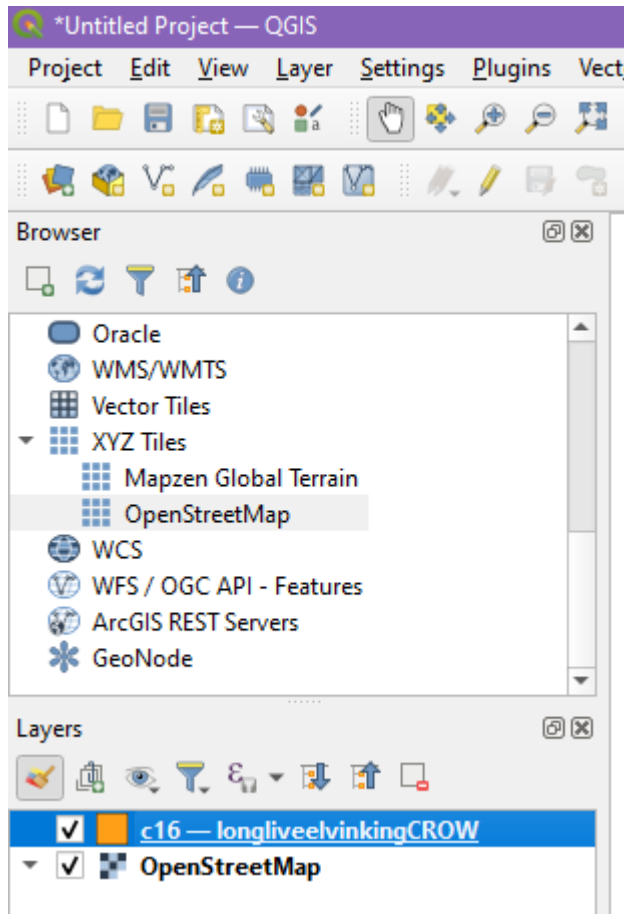


Figure 16-4. QGIS Overlays.

4. Now right-click on the imported GeoJson data, i.e., **c16 – longliveelvinkingCROW** and go to properties -> Source. Here you can find that the default Coordinate Reference System (CRS) is set to **EPSG: 4326 – WGS 84**.

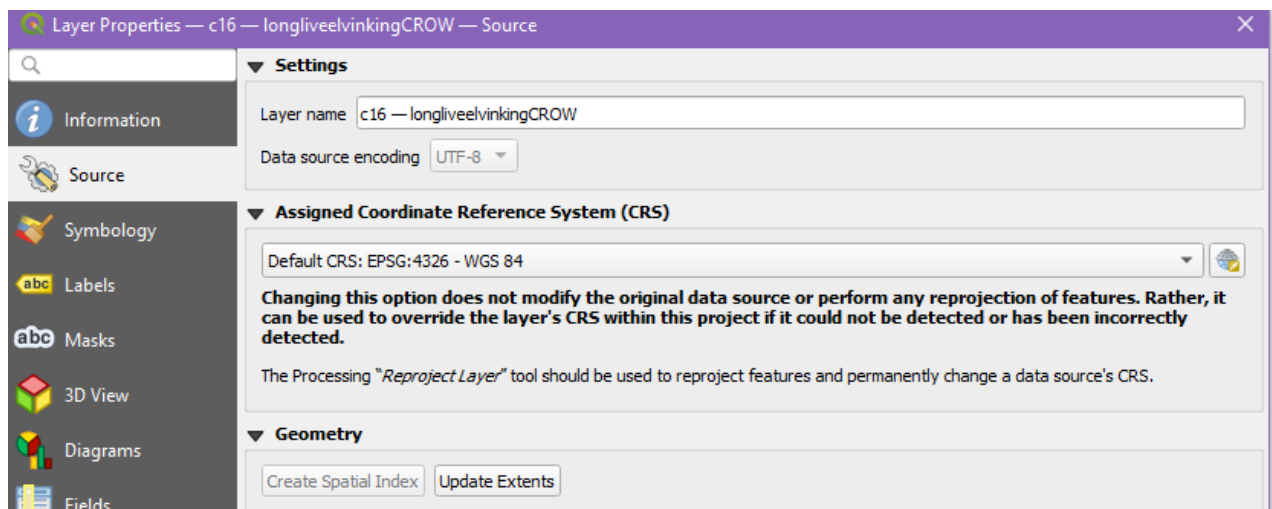


Figure 16-5. QGIS Default Coordinate System.

5. Upon analysis of the GeoJson data coordinates and the challenge description in Figure 16-1, You can understand that the default coordinate system cannot visualise the coordinates. Set the Coordinate Reference System (CRS) to **EPSG:2193 – NZGD 2000 / New Zealand Transverse Mercator 2000**

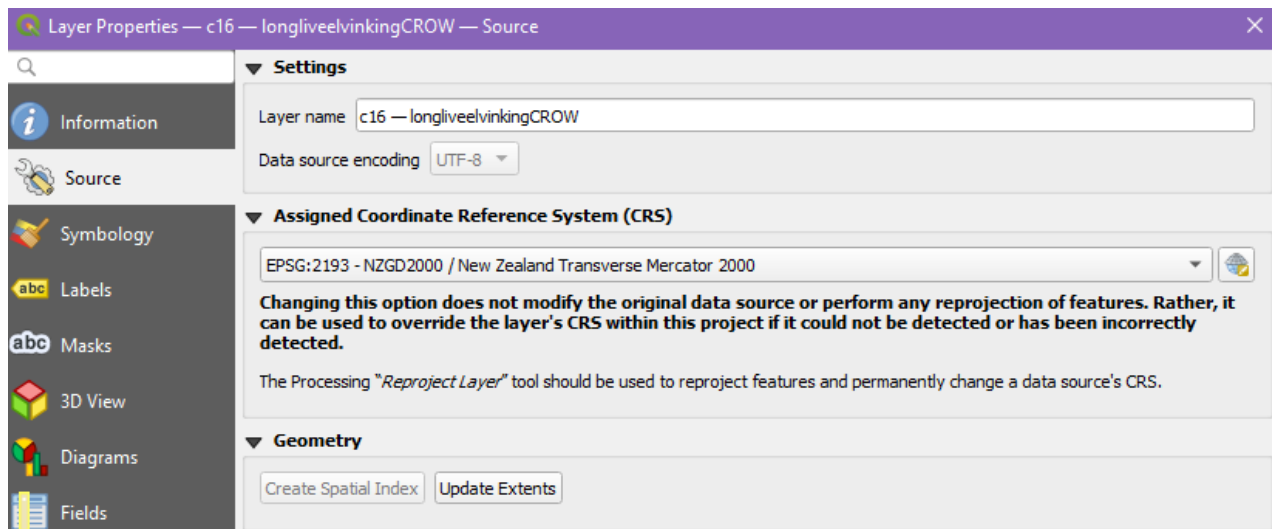


Figure 16-6. QGIS New Zealand Coordinate System.

6. Finally, The GeoJson data can be visualised in the QGIS software, which looks like the following:



Figure 16-7. QGIS Overlays View.

7. After further zooming, we get the flag as the following:



Figure 16-8. The Final Flag.

FLAG: RUNE{ITIUTEVUYLXAUTLF}