

The_Chemistry_Of_Code


Description

CHALLENGE

210 SOLVES



The Chemistry Of Code

 200

REVERSE

During a discussion with hackers, one idea stuck: every hacker must be able to understand code, no matter how cryptic.

Hidden within obfuscated **Rust** code is a function that holds the flag. But it only unlocks when you reverse its logic and uncover the correct username and password.

Do you have what it takes to decode this tangled masterpiece? Dive into the chemistry of code and prove your skills.

Hint: Some secrets are worth their salt.



chemistryof...

Flag

Submit

Category

#reverse

Solution

Extracting the file, we see a rust file.

```
(atlas@kali)-[~/Desktop/chemistryofcode/src]
$ cat main.rs
use base64::{engine::general_purpose::STANDARD, Engine};
use hex::encode as hex_encode;
use num_bigint::BigUint;
use std::io::{self, Write};

const FERROUS_OXIDE_USERNAME: &str = "AdminFeroxide";
const ANIONIC_PASSWORD: &str = "NjQzMzcyNzUzNTMzMzE2Njc5MzE2ZTM2";
const ALKALINE_SECRET: &str = "4143454354467B34707072336E373163335F3634322C28010D3461302C392E";

fn ionic_bond(cation_input: &str, anion_input: &str) {
    let cation_hex = hex_encode(cation_input);
    let anion_hex = hex_encode(anion_input);

    let cation_value = BigUint::parse_bytes(cation_hex.as_bytes(), 16).unwrap();
    let anion_value = BigUint::parse_bytes(anion_hex.as_bytes(), 16).unwrap();

    let covalent_link = &cation_value ^ &anion_value;

    let alkaline_secret_value = BigUint::parse_bytes(ALKALINE_SECRET.as_bytes(), 16).unwrap();
    let metallic_alloy = &covalent_link ^ &alkaline_secret_value;

    let precipitate = format!("{:x}", metallic_alloy);

    let alloy_compound = (0..precipitate.len())
        .step_by(2)
        .map(|i| u8::from_str_radix(&precipitate[i..i + 2], 16).unwrap() as char)
        .collect::<String>();

    println!("Crystallized Flag (ASCII): {}", alloy_compound);
}
```

There is a password and a secret with the latter is more likely to be a flag since the password is a base64 string.

Decoding the password gives us a hex string, the 'e' is a giveaway.

Decode from Base64 format


Simply enter your data then push the decode button.



NjQzMzcyNzUzNTM3MzE2Njc5MzE2ZTM2

 For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8  Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

 Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

 **DECODE**  Decodes your data into the area below.

643372753537316679316e36

Decoding Hex to ASCII is possible, so we might use that.

From


Hexadecimal

▼


To

Text

▼

 Open File

Sample



Paste hex code numbers or drop file

643372753537316679316e36

Character encoding

ASCII

▼

= Convert

× Reset

↕ Swap

d3ru571fy1n6

When we run the rust file, it asks for a catalyst and a reagent, similar to username/password.

(atlas@kali)-[~/Desktop/chemistryofcode/src]

\$ cargo run

```
Updating crates.io index /-regular chemistryofcode.zip exploit-me
Downloaded autocfg v1.4.0 /ovpn
Downloaded hex v0.4.3
Downloaded num-integer v0.1.46 "Brokenfr (2)" selected (containing 1
Downloaded num-traits v0.2.19
Downloaded base64 v0.22.1
Downloaded num-bigint v0.4.6
Downloaded 6 crates (289.4 KB) in 0.50s
Compiling autocfg v1.4.0
Compiling hex v0.4.3
Compiling base64 v0.22.1
Compiling num-traits v0.2.19
Compiling num-integer v0.1.46
Compiling num-bigint v0.4.6
Compiling chemistryofcode v0.1.0 (/home/atlas/Desktop/chemistryofcode)
Finished `dev` profile [unoptimized + debuginfo] target(s) in 6.91s
Running /home/atlas/Desktop/chemistryofcode/target/debug/chemistryofcode`
```

Introduce the Catalyst: AdminFeroxide

Introduce the Reagent: d3ru571fy1n6

Crystallized Flag (ASCII): ACECTF{4ppr3n71c3_w4l73r_wh1t3}