1. Gambar Berita (Image.png)
Diberikan sebuah link, cache.pusatkode.com/dI8dAsWyJsQW, diberikan sebuah file tanpa ekstensi, saya cek menggunakan HxD tertulis “PNG”, dan ternyata benar itu merupakan sebuah gambar berita.

01 JANUARI 2026

Berita Rakyat Indonesia

Sidang perdana kasus peretasan perbankan terbesar dalam sejarah Indonesia yang melibatkan terdakwa berinisial RX berakhir dengan ketegangan hebat setelah ia melontarkan ancaman sabotase finansial Indonesia dari kursi Negeri Pengadilan Masyarakat.

Setelah didakwa atas pencurian dana senilai Rp500 miliar dari lima bank papan atas, RX secara terbuka menyatakan bahwa timnya masih menguasai akses backdoor aktif di puluhan institusi keuangan Indonesia yang siap memicu perampokan digital masif secara otomatis saat ia mulai menjalani masa tahanan. Terdapat sebuah pesan kriptik yang diduga sebagai langkah krusial menuju password laptop sang peretas:

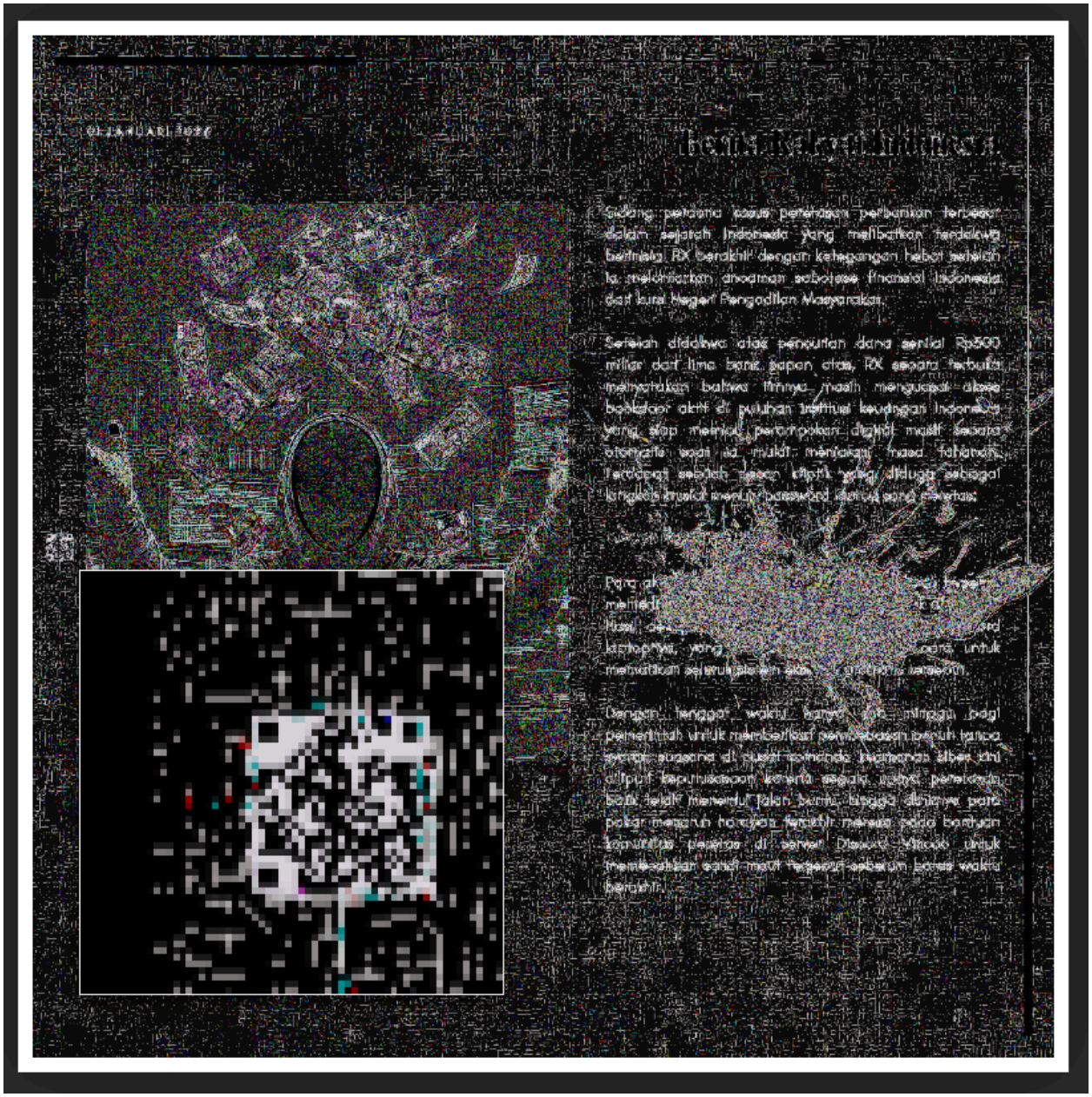
QECGDELIC 'FE'HS

Para ah[illegible] memerl[illegible] Hasil d[illegible] laptopnya, yang [illegible] mematikan seluruh sistem eks[illegible] tersebut.

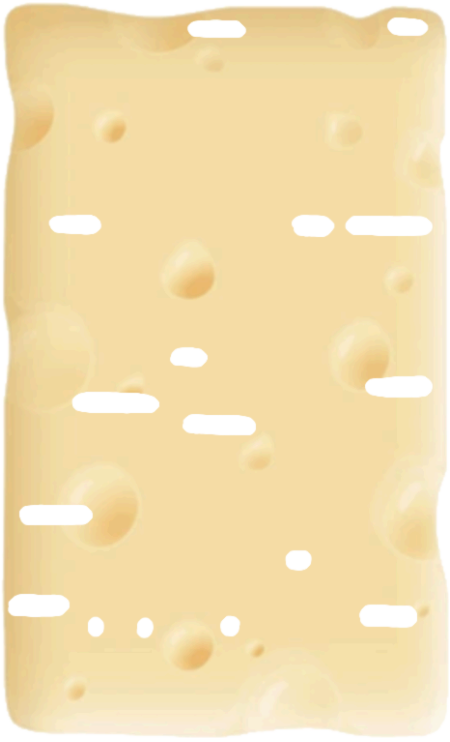
Dengan tenggat waktu hanya [illegible] minggu bagi pemerintah untuk memberikan pembebasan penuh tanpa syarat, suasana di pusat komando keamanan siber kini diliputi keputusasaan karena segala upaya peretasan balik telah menemui jalan buntu, hingga akhirnya para pakar menaruh harapan terakhir mereka pada bantuan komunitas peretas di server Discord Vincoo untuk memecahkan sandi maut tersebut sebelum batas waktu berakhir.

Dua Minggu Menuju Sabotase: Nasib Perbankan Indonesia

Berbagai cara pengecekan steganography saya coba melalui https://29a.ch/photo-forensics/#noise-analysis, Pada method Noise analysis terlihat QR Kecil pada pojok kiri.



2. QR dalam gambar berita discan mengarah pada link yang mengarahkan ke https://cache.pusatkode.com/3HPhwvZVs0X9, saya mencoba mengubah ext ke PNG seperti link awal, terbukalah sebuah gambar keju bolong.



3. Awalnya saya bingung apa fungsinya, saya melakukan pengecekan standard CTF kembali pada gambar tersebut tapi nihil, namun ukuran gambarnya yang kecil membuat saya bertanya-tanya, karena sangat berbanding terbalik dengan file pertama yang besar. Pada channel diskusi ada seseorang yang membahas tentang Cheese of truth, akhirnya saya mencoba menggabungkan gambar berita + keju, pada posisi text berita. Awalnya hal tersebut tidak menunjukan apapun, setelah dilakukan flip gambar pada keju secara vertikal, terbaca tulisan jelas N P M RX Backdoor password.

4. Karena kita tahu NPM itu merupakan Node Package Manager, langsung saja saya search sebuah package di https://www.npmjs.com/ dengan keyword rxbackdoorpassword. Disini saya menemukan sebuah package https://www.npmjs.com/package/rxbackdoorpassword/v/1.0.6 Dengan author account1pusatkode. Saya yakin ini langkah yang benar karena kita tahu bahwa author tersebut sama dengan link yang diberikan “pusatkode”. Disini saya lanjut untuk analysis terhadap kode yang ada di package tersebut.

index.js

```
var c=(e=>typeof require<"u"?require:typeof Proxy<"u"?new Proxy(e,{get:(n,t)=>(typeof require<"u"?require:n)[t]}):e)(function(e){if(typeof require<"u")return require.apply(this,arguments);throw Error('Dynamic require of "'+e+'" is not supported')});var o=c("crypto"),s="eb68eed0f54f5d12a3f6911e14f61598d0be15ba44579d55659ec23e49f43c01e302e6e2f15d781d95e8a44b2c9c11dfe2db0dbe6a7d9d4864a0b13e43f7065ed668fad2",f=Buffer.from(s,"hex");function a(e,n){let t=Buffer.alloc(e.length);for(let r=0;r<e.length;r++)t[r]=e[r]^n[r%n.length];return t}function d(e){return a(f,o.createHash("sha256").update(String(e)).digest()).toString("utf8")}(async function(){let n=process.argv.slice(2),t=Number(n[0]);if(!Number.isInteger(t))return process.exit(0);let r=d(t);console.log(r)})();
```

Kode JavaScript ini adalah program CLI Node.js sederhana untuk mendekripsi pesan tersembunyi menggunakan XOR dengan kunci hash SHA-256. Program memuat modul crypto, lalu menyimpan sebuah data terenkripsi dalam bentuk hex string yang diubah menjadi Buffer. Fungsi a() melakukan operasi XOR byte-per-byte antara data terenkripsi dan sebuah kunci. Kunci tersebut dihasilkan oleh fungsi d(), yaitu hash SHA-256 dari angka input (argumen CLI) yang kemudian dipakai sebagai key XOR. Saat program dijalankan, ia mengambil satu argumen angka dari command line, memastikan input tersebut integer, lalu mendekripsi data dengan XOR menggunakan hash angka tersebut dan mencetak hasilnya sebagai teks UTF-8. Dengan kata lain, angka input adalah kunci dekripsi, dan hanya angka yang benar akan menghasilkan pesan terbaca.

Disini saya langsung saja membuat solver untuk melakukan bruteforce terhadap encrypted hex yang ditemukan pada index.js

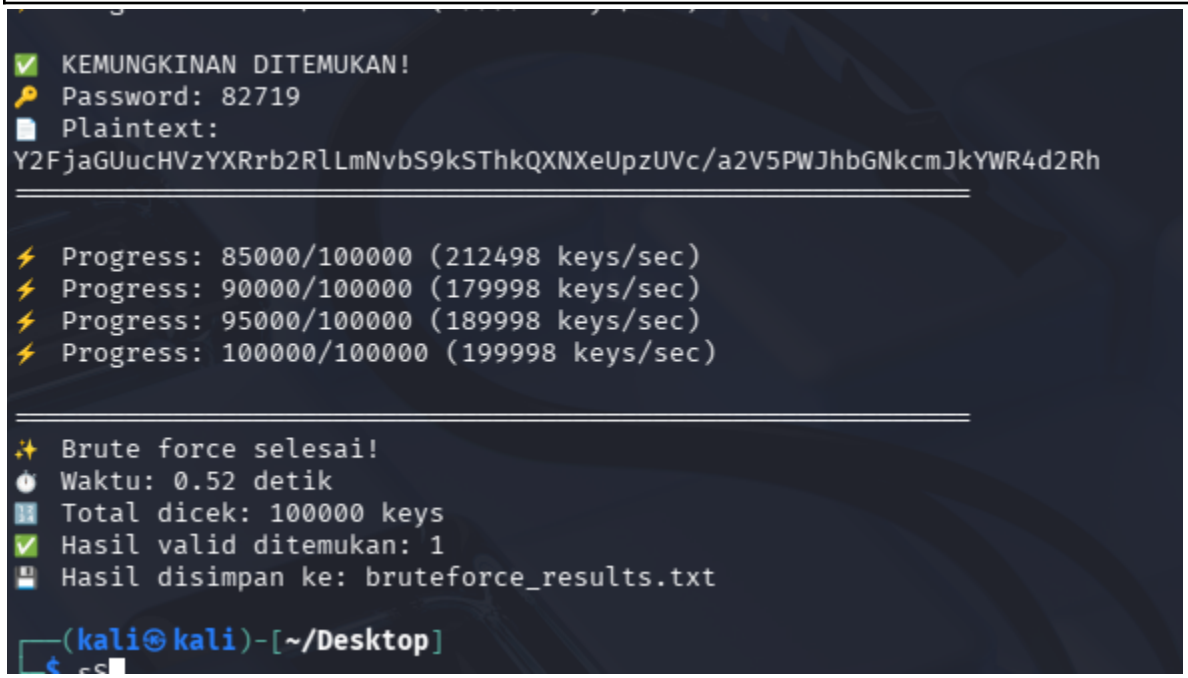
brute.js

```
const crypto = require('crypto');
const fs = require('fs');

// Ciphertext dari package
const cipherHex =
"eb68eed0f54f5d12a3f6911e14f61598d0be15ba44579d55659ec23e49f43c01e302e6e2f15d7
81d95e8a44b2c9c11dfe2db0dbe6a7d9d4864a0b13e43f7065ed668fad2";
const ciphertext = Buffer.from(cipherHex, "hex");

// Fungsi XOR
function xorDecrypt(data, key) {
  let result = Buffer.alloc(data.length);
  for (let i = 0; i < data.length; i++) {
    result[i] = data[i] ^ key[i % key.length];
  }
  return result;
}

// Fungsi decrypt dengan angka
function decrypt(num) {
  const hash = crypto.createHash("sha256").update(String(num)).digest();
  return xorDecrypt(ciphertext, hash).toString("utf8");
}

// Fungsi cek apakah hasil valid (readable text)
function isValidText(text) {
  // Cek apakah sebagian besar karakter adalah printable ASCII atau UTF-8
  let validChars = 0;
  for (let i = 0; i < text.length; i++) {
    const code = text.charCodeAt(i);
    // Printable ASCII (32-126) atau newline/tab
    if ((code >= 32 && code <= 126) || code === 10 || code === 13 || code === 9) {
```

```
      validChars++;
    }
  }
  // Jika >80% karakter valid, kemungkinan plaintext yang benar
  return (validChars / text.length) > 0.8;
}

// Brute force function
function bruteForce(start = 1, end = 100000) {
  console.log(`🔍 Memulai brute force dari ${start} sampai ${end}...`);
  console.log(`⏱️  Estimasi waktu: ~${Math.ceil((end - start) / 10000)} detik\n`);

  const results = [];
  const startTime = Date.now();
  let checked = 0;

  for (let i = start; i <= end; i++) {
    try {
      const plaintext = decrypt(i);

      // Progress indicator setiap 5000 iterasi
      if (i % 5000 === 0) {
        const elapsed = ((Date.now() - startTime) / 1000).toFixed(1);
        const rate = (i - start) / elapsed;
        console.log(`⚡ Progress: ${i}/${end} (${rate.toFixed(0)} keys/sec)`);
      }

      // Cek apakah hasil valid
      if (isValidText(plaintext)) {
        console.log(`\n✅ KEMUNGKINAN DITEMUKAN!`);
        console.log(`🔑 Password: ${i}`);
        console.log(`📄 Plaintext:\n${plaintext}`);
        console.log(`${'='.repeat(60)}\n`);

        results.push({
          password: i,
          plaintext: plaintext
        });
      }

      checked++;
    } catch (e) {
      // Skip jika error
    }
  }

  const totalTime = ((Date.now() - startTime) / 1000).toFixed(2);

  console.log(`\n${'='.repeat(60)}`);
  console.log(`✨ Brute force selesai!`);
```

```
  console.log(`⏱️  Waktu: ${totalTime} detik`);
  console.log(`🔢 Total dicek: ${checked} keys`);
  console.log(`✅ Hasil valid ditemukan: ${results.length}`);

  // Simpan hasil ke file
  if (results.length > 0) {
    const output = results.map(r =>
      `Password: ${r.password}\n${'='.repeat(40)}\n${r.plaintext}\n${'='.repeat(60)}\n`
    ).join('\n');

    fs.writeFileSync('bruteforce_results.txt', output);
    console.log(`💾 Hasil disimpan ke: bruteforce_results.txt`);
  }

  return results;
}

// Fungsi test angka spesifik
function testSpecific(numbers) {
  console.log(`🧪 Testing angka spesifik: ${numbers.join(', ')}\n`);

  numbers.forEach(num => {
    try {
      const plaintext = decrypt(num);
      console.log(`\n🔑 Password: ${num}`);
      console.log(`Valid: ${isValidText(plaintext) ? '✅' : '❌'}`);
      console.log(`Output:\n${plaintext}`);
      console.log(`${'='.repeat(60)}`);
    } catch (e) {
      console.log(`❌ Error dengan ${num}`);
    }
  });
}

// MAIN EXECUTION
const args = process.argv.slice(2);

if (args.length === 0) {
  // Default: brute force 1-100000
  bruteForce(1, 100000);
} else if (args[0] === 'test') {
  // Mode test dengan angka spesifik
  const testNumbers = [
    143,      // sum semua huruf
    19,       // S (huruf terakhir)
    72, 27,   // skip pattern
    14163,    // NPM concat
    3095238,  // IP dari README
    1416, 1613, 1413,  // NPM pairs
    64819,    // modulo result
```

```
    21256,    // first digit pattern
  ];
  testSpecific(testNumbers);
} else if (args.length === 2) {
  // Custom range: node bruteforce.js 1000 5000
  const start = parseInt(args[0]);
  const end = parseInt(args[1]);
  bruteForce(start, end);
} else {
  // Test single number
  const num = parseInt(args[0]);
  testSpecific([num]);
}
```

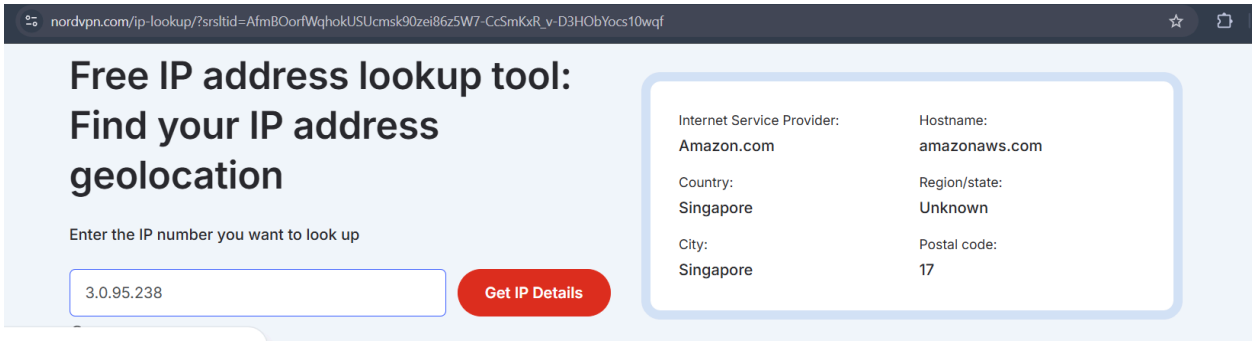


Terdapat sebuah base64 di angka 82719:
Y2FjaGUucHVzYXRrb2RlLmNvbS9kSThkQXNXeUpzUVc/a2V5PWJhbGNkcmJkYWR4d2Rh
Ketika kita lakukan decode kita mendapatkan sebuah link:
cache.pusatkode.com/dI8dAsWyJsQW?key=balcdrbdadxwda

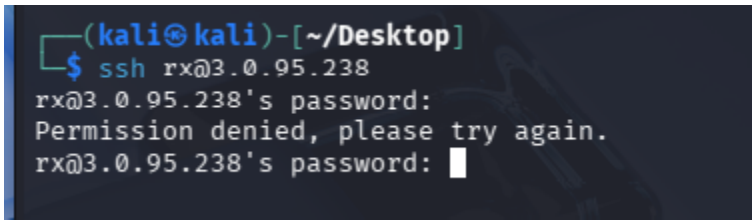
Disini ketika saya akses akan mendownload gambar berita yang awal. Disini saya sempat stuck seharian gk tidur gk makan :v bingung mau ngapain lagi.

Lalu ketika saya traceback terdapat IP di history version NPM package nya. Disini terdapat 47.129.57.16 pada version v1.0.5 dan 3095238 pada version v1.0.6. Ketika dilakukan testing IP 47.129.57.16 tidak mendapatkan response apa apa (timed out). Disini saya stuck lagi, saya bingung angka ini tuh sebener nya apa (3095238) kalo dipikir-pikir apakah ini sebuah port? Tapi port hanya limitnya 65000-an. Disini saya coba coba angka 3095238 apakah ini merupakan IP?

TERNYATA ketika saya cek di https://nordvpn.com/ip-lookup ternyata valid, 3.0.95.238 merupakan IPnya.



Langsung saja saya coba akses menggunakan ssh dengan user rx

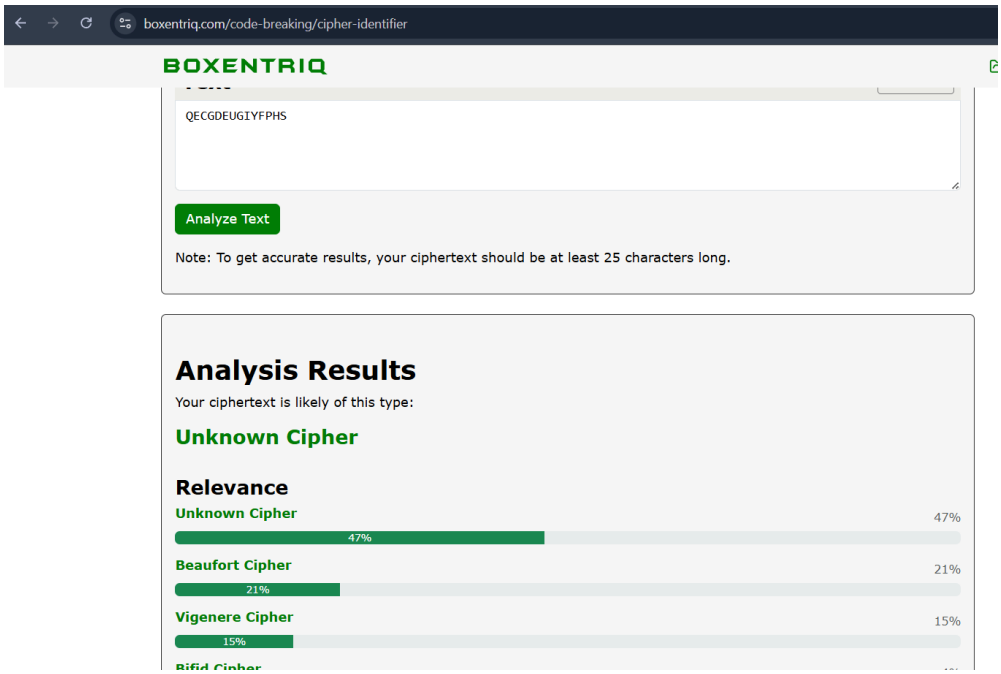


Disini saya stuck lagi. Kita harus mendapatkan sebuah password untuk akun tersebut.

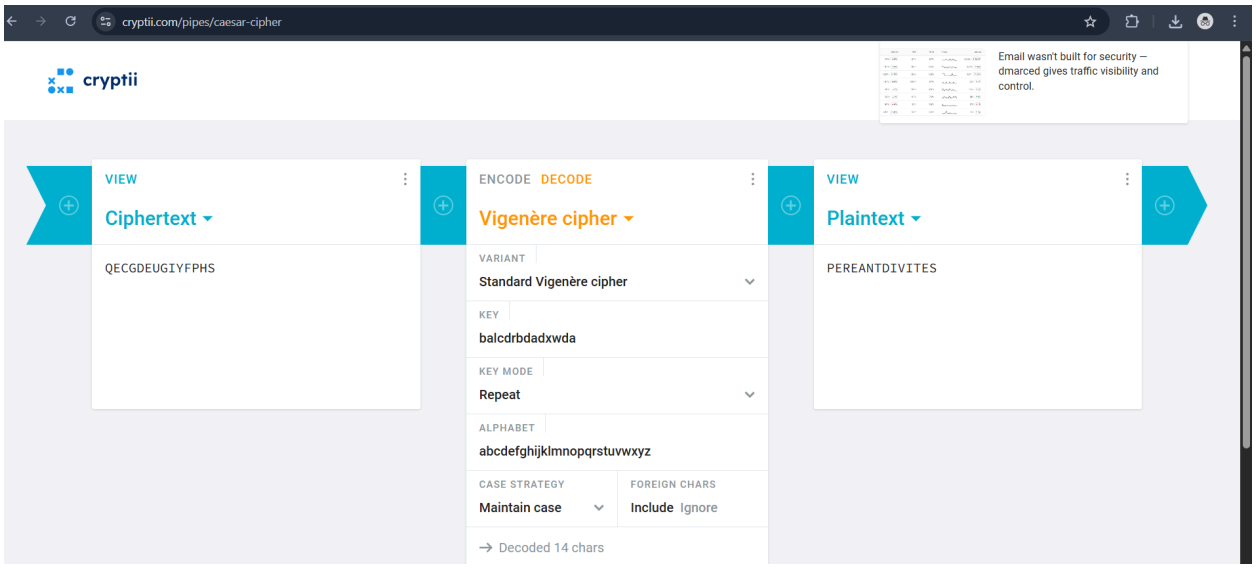
Informasi yang kita ketahui kita sudah mendapatkan informasi sebagai berikut:

- Pesan kriptik : QECGDEUGIYFPHS
- Sebuah kunci : balcdrbdadxwda

Disini saya coba untuk melakukan Cipher Identifier terhadap pesan kriptik tersebut.



Ternyata kemungkinan besar ini merupakan viginere cipher, dan kita tahu bahwa viginere ini membutuhkan key. Langsung saja saya coba untuk decrypt dengan informasi yang ada di website ini:



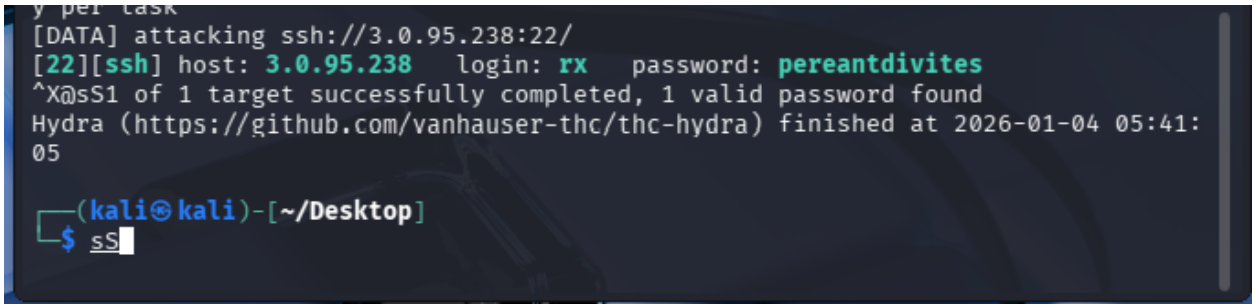
Dan kita mendapatkan sebuah kata??
PEREANTDIVITES yang jika dicari pada google, ternyata merupakan bahasa latin, yang sangat sesuai dengan tema kali ini.

The Latin phrase "**PEREANT DIVITES**" translates to "**Let the rich perish**".

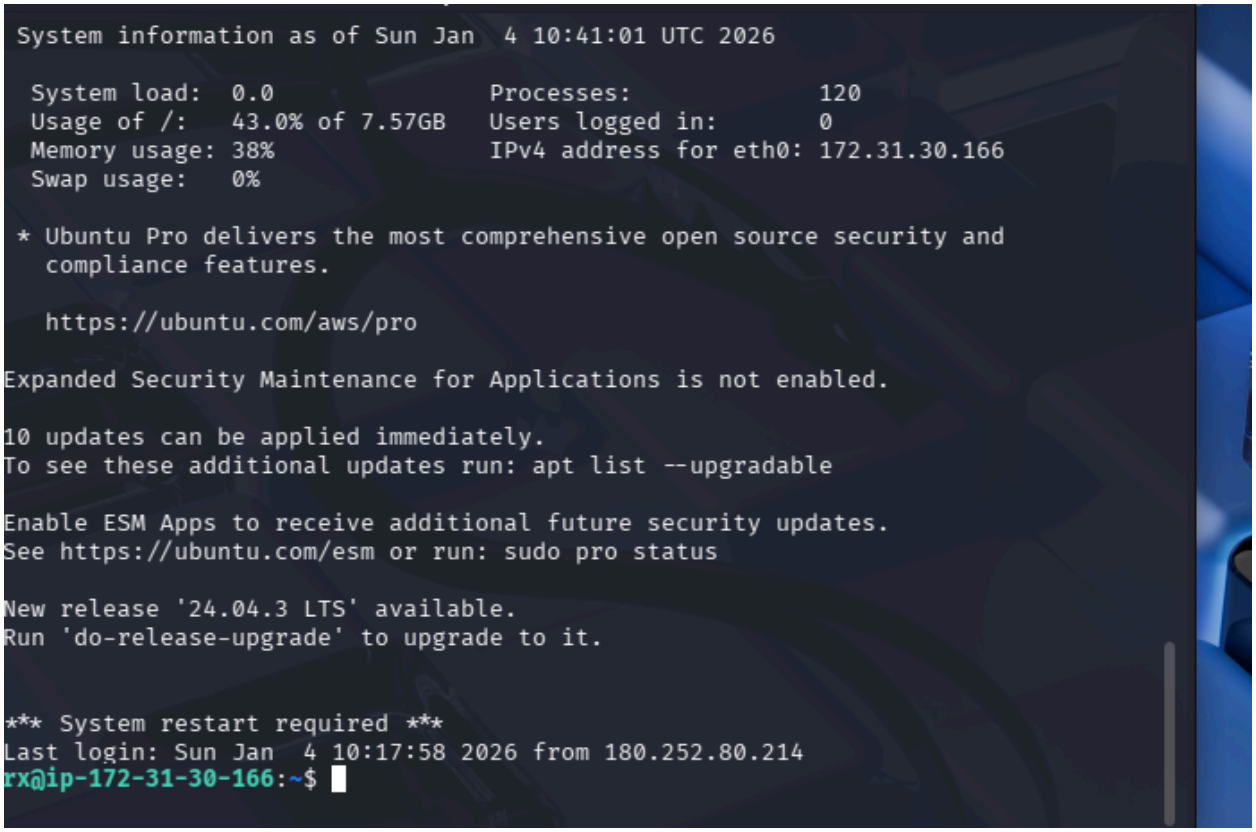
The phrase is a combination of:

- ***Pereant***: The third-person plural present active subjunctive of the verb *pereō*, meaning "to perish, die, be destroyed, go to ruin". The subjunctive mood here expresses a wish, command, or imprecation.
- ***Divites***: The nominative plural of the adjective/noun *dives*, meaning "rich" or "wealthy people".

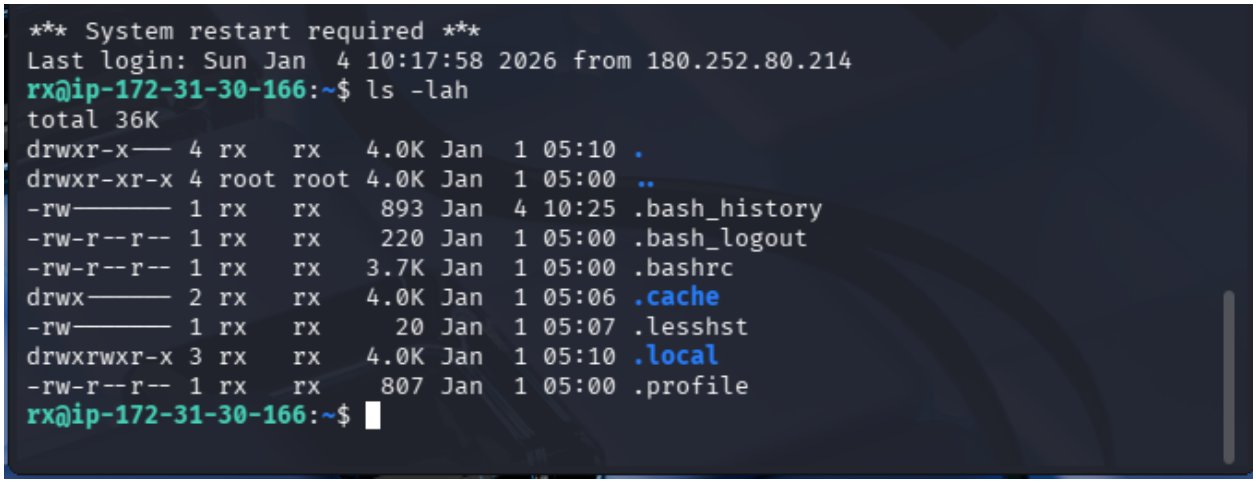
6. Coba akses SSH rx dengan kata yang ditemukan, karena format awal yang menggunakan semua huruf besar tidak bisa, saya mencoba berbagai format menggunakan hydra.



Sampai sini langsung saja saya coba akses ssh tersebut dan BERHASIL!!!! Ini merupakan password nya!!!!



Disini saya coba untuk ngecek list file menggunakan ls -lah



Hmm tidak ada yang mencurigakan, Tapi coba saya cek satu satu mulai dari .bash_history.

```
rx@ip-172-31-30-166:~$ cat .bash_history | head -20
service ls
systemctl status tor
clear
ls
cd ~
ls
cd /root
cd /var
systemctl reload nginx
groups rx
groups root
groups rootl
nano /etc/nginx/sites-available/tor-redirect
nano /etc/tor/torrc~
nano /etc/tor/torrc
exit
whoami
ls
ls -lah
cat .bash_history
rx@ip-172-31-30-166:~$
```

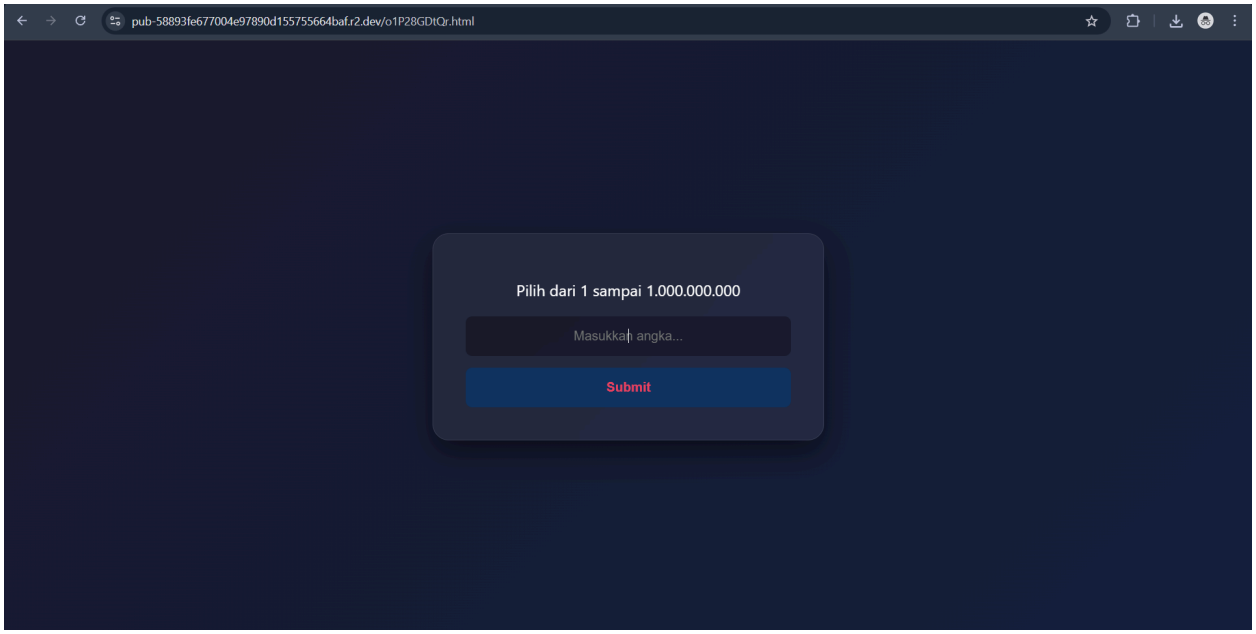
Ternyata user rx ini sebelumnya melakukan suatu modifikasi pada file berikut
/etc/nginx/sites-available/tor-redirect
/etc/tor/torrc

Disini saya coba cek file /etc/nginx/sites-available/tor-redirect

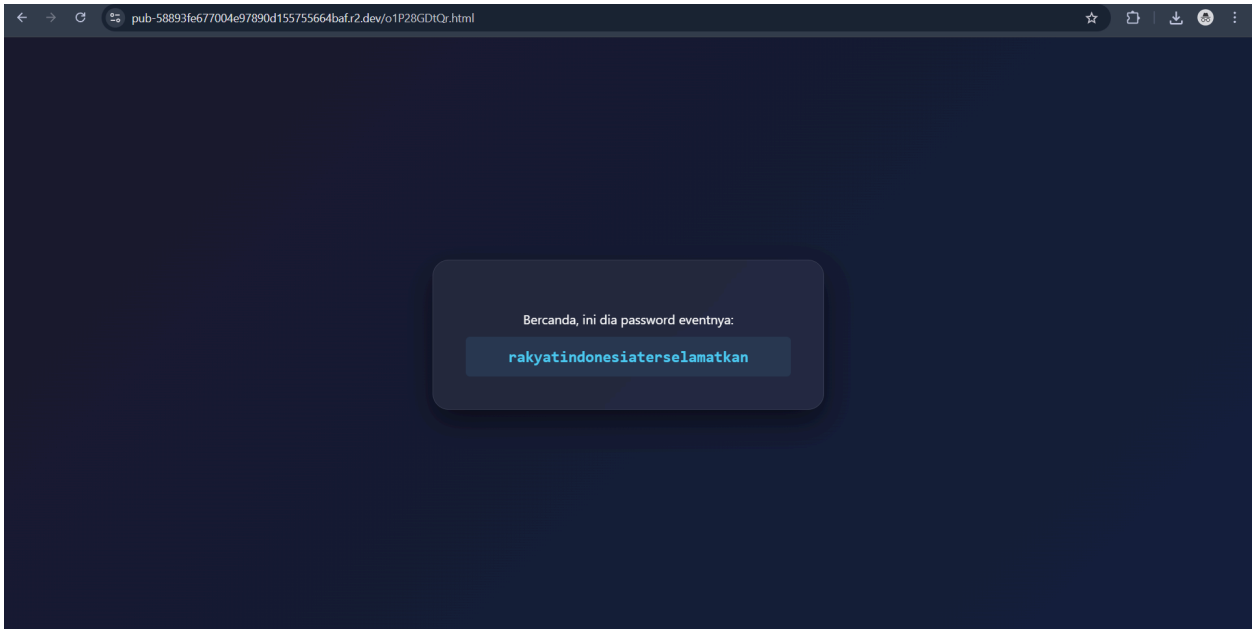
```
rx@ip-172-31-30-166:~$ cat /etc/nginx/sites-available/tor-redirect
server {
    listen 127.0.0.1:8080;
    server_name _;

    location / {
        return 302 https://cache.pusatkode.com/o1P28GDtQr.html;
    }
}
```

Ini merupakan konfigurasi nginx dimana ketika kita mengakses 127.0.0.1:8080 akan di forward ke url https://cache.pusatkode.com/o1P28GDtQr.html

Langsung saja saya cek url tersebut



Terdapat sebuah form dimana kita harus memasukan password yang benar! Disini saya terlintas sebuah angka yang kita dapatkan pada saat melakukan bruteforce, 82719!



DAN KITA SUDAH MENDAPATKAN PASSWORD YANG BENAR!!!!!