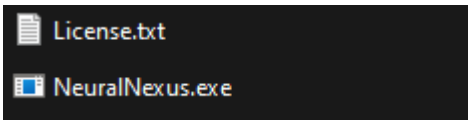
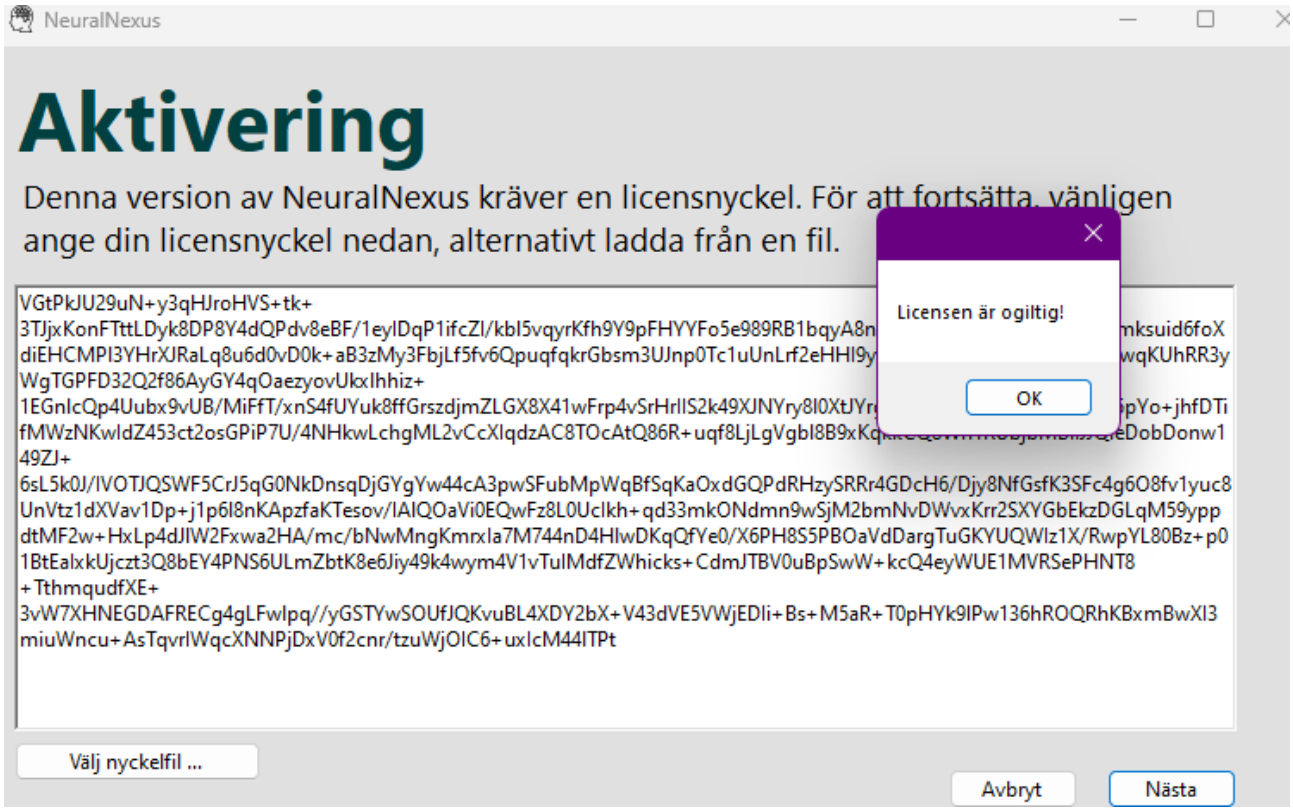
NeuralNexus

av Challeballe

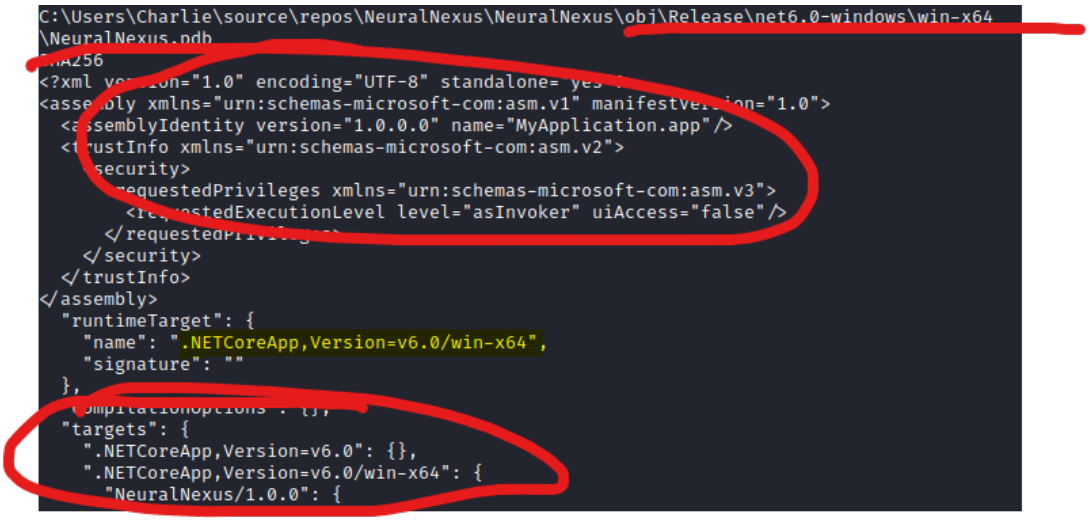
Du möts av två filer, en binär och en txt fil.



Startar man progammet möts man av ett inputfält där man måste ange en licens. Testar vi med licensen vi fått medskickad så är denna ogiltig.



Kör vi strings på appen så ser vi gaaanska snabbt att vi har en C# .NET fil här.



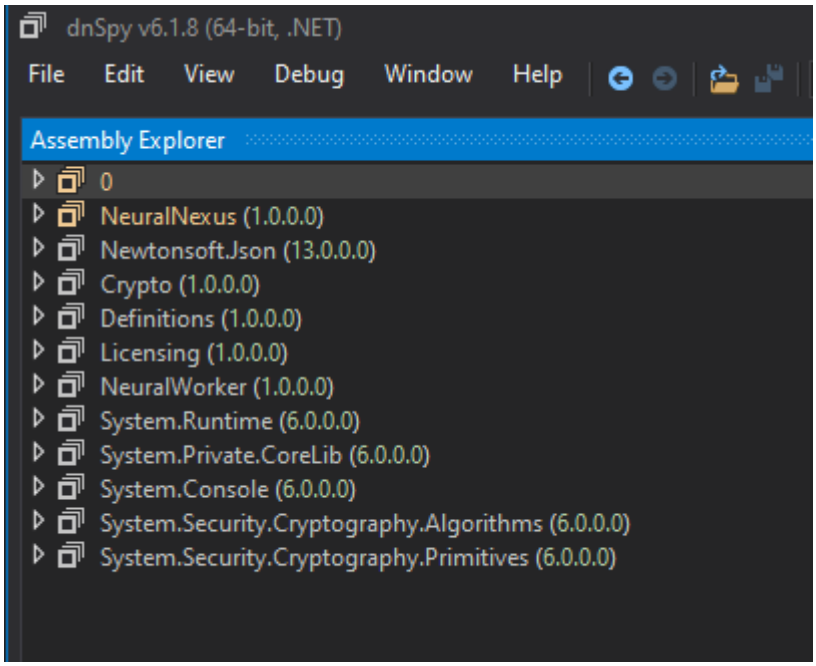
Vi ser också massa referenser till diverse DLLer som verkar lite intressanta:

```
  },
  "Crypto/1.0.0": {
    "runtime": {
      "Crypto.dll": {}
    }
  },
  "Definitions/1.0.0": {
    "runtime": {
      "Definitions.dll": {}
    }
  },
  "Licensing/1.0.0": {
    "dependencies": {
      "Crypto": "1.0.0",
      "Newtonsoft.Json": "13.0.3"
    },
    "runtime": {
      "Licensing.dll": {}
    }
  },
  "NeuralWorker/1.0.0": {
    "dependencies": {
      "Definitions": "1.0.0"
    },
    "runtime": {
      "NeuralWorker.dll": {}
    }
  }
```

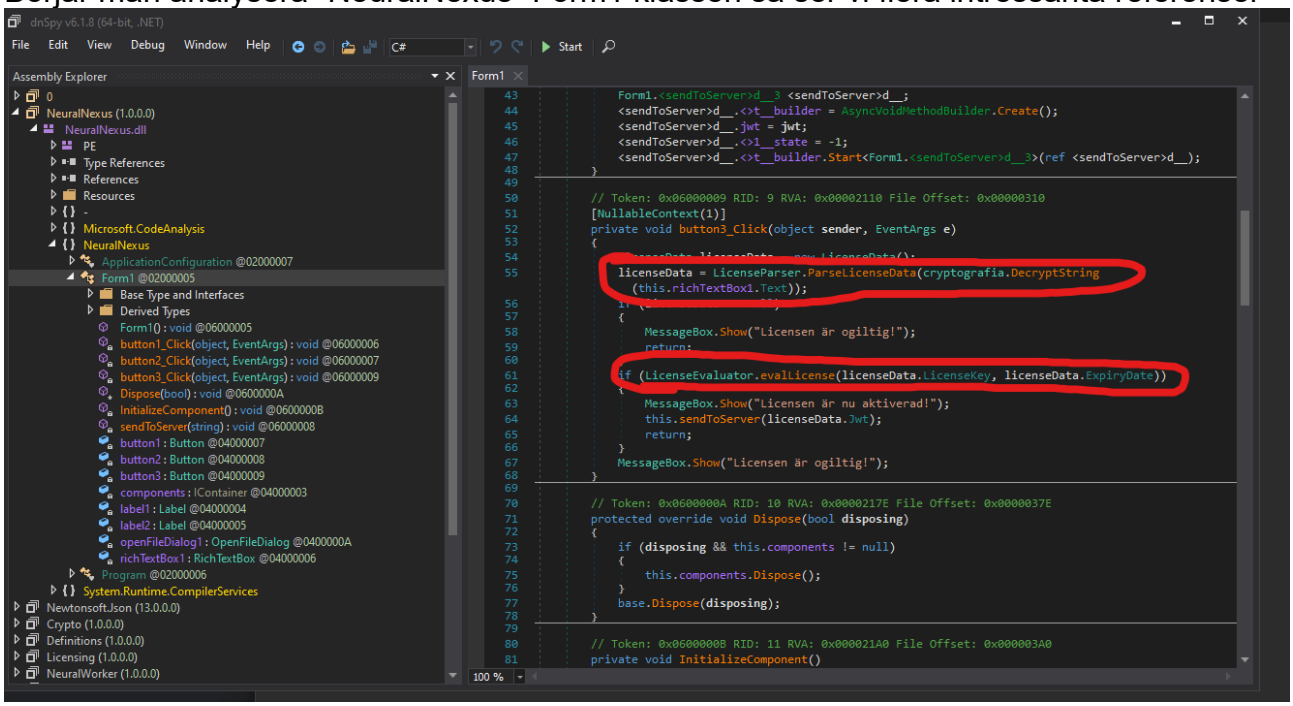
Eftersom vi bara fick en exe-fil och inga DLLer så får vi hoppas att dessa är inbakade. Vi kan få ut alla separata filer med en gammal hedelig binwalk:

```
└─$ binwalk --dd='exe' NeuralNexus.exe

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Microsoft executable, portable (PE)
146295        0x23B77         XML document, version: "1.0"
147968        0x24200         Microsoft executable, portable (PE)
156876        0x264CC         PNG image, 256 x 256, 8-bit/color RGBA, non-interlaced
156917        0x264F5         Zlib compressed data, best compression
269687        0x41D77         XML document, version: "1.0"
272976        0x42A50         Microsoft executable, portable (PE)
812476        0xC65BC         Copyright string: "CopyrightAttribute"
973400        0xEDA58         Object signature in DER format (PKCS header length: 4, seq
uence length: 12031
973541        0xEDAE5         Certificate in DER format (x509 v3), header length: 4, seq
uence length: 1380
974925        0xEE04D         Certificate in DER format (x509 v3), header length: 4, seq
uence length: 1680
976609        0xEE6E1         Certificate in DER format (x509 v3), header length: 4, seq
uence length: 2015
979505        0xEF231         Object signature in DER format (PKCS header length: 4, seq
uence length: 5926
979674        0xEF2DA         Certificate in DER format (x509 v3), header length: 4, seq
uence length: 1728
981406        0xEF99E         Certificate in DER format (x509 v3), header length: 4, seq
uence length: 1710
983120        0xF0050         Certificate in DER format (x509 v3), header length: 4, seq
uence length: 1421
985440        0xF0960         Microsoft executable, portable (PE)
992096        0xF2360         Microsoft executable, portable (PE)
996704        0xF3560         Microsoft executable, portable (PE)
1002848       0xF4D60         Microsoft executable, portable (PE)
```

Eftersom det är en .NET app så kan vi dra in alla filer i dnSpy för att få ut källkoden:



Börjar man analysera “NeuralNexus” Form1 klassen så ser vi flera intressanta referenser



Kollar vi på LicenseEvaluator klassen hittar vi funktionen som verkar hantera om licensen är gilltig:

```
public class LicenseEvaluator
{
    // Token: 0x0600000E RID: 14 RVA: 0x00002118 File Offset: 0x00000318
    [NullableContext(1)]
    public static bool evalLicense(string license, DateTime exp)
    {
        return !(exp < DateTime.Now) && HashUtility.ComputeSha256Hash(license) ==
          "10c8e5217f0128dedb2c843fa82e933d4bbc0cc5d513c30071ac496996249947";
    }
}
```

Så länge datumet inte är tidigare än just nu och hashen matchar så verkar det tolkas som en gilltig licens.

Det som skickas in i den funktionen är resultatet från LicenseParser klassens funktion “ParseLicenseData” vilket bara är en json deserialiserare:

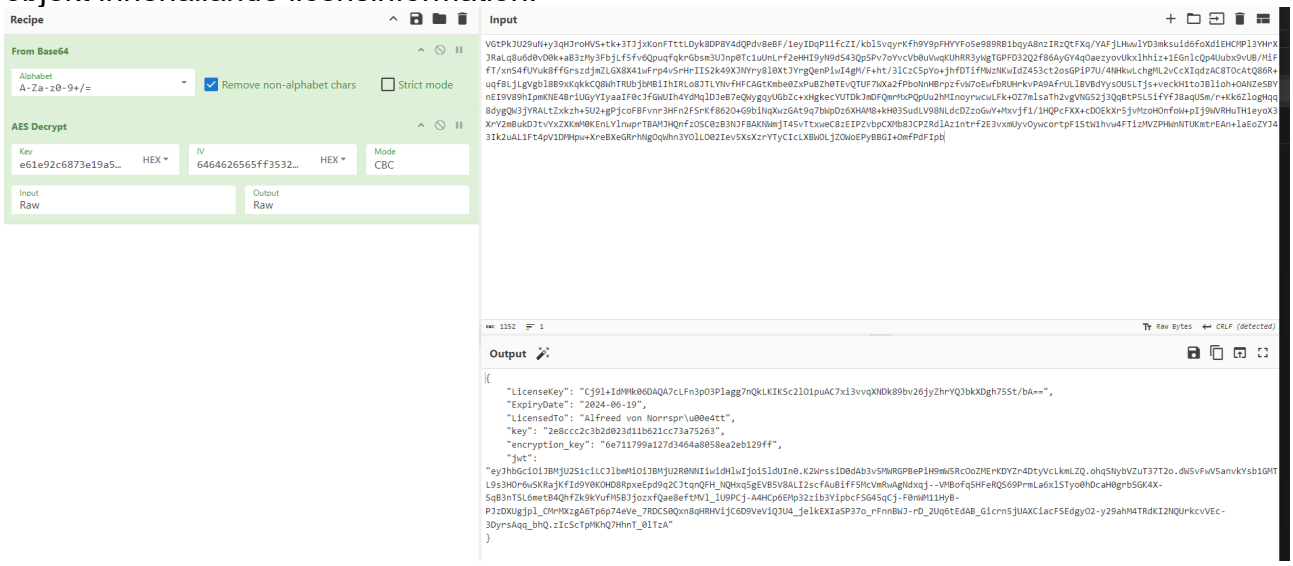
```
public static LicenseData ParseLicenseData(string json)
{
    LicenseData result;
    try
    {
        result = JsonConvert.DeserializeObject<LicenseData>(json);
    }
    catch (Exception ex)
    {
        Console.WriteLine("An error occurred: " + ex.Message);
        result = null;
    }
    return result;
}
```

Den kanske mest intressanta klassen i dethär fallet är “cryptografia” som har funktioner för att avkryptera licensfilen. Längst ner i den klassen hittar vi både nyckeln och IVn som krävs för att avkryptera licensfilen:

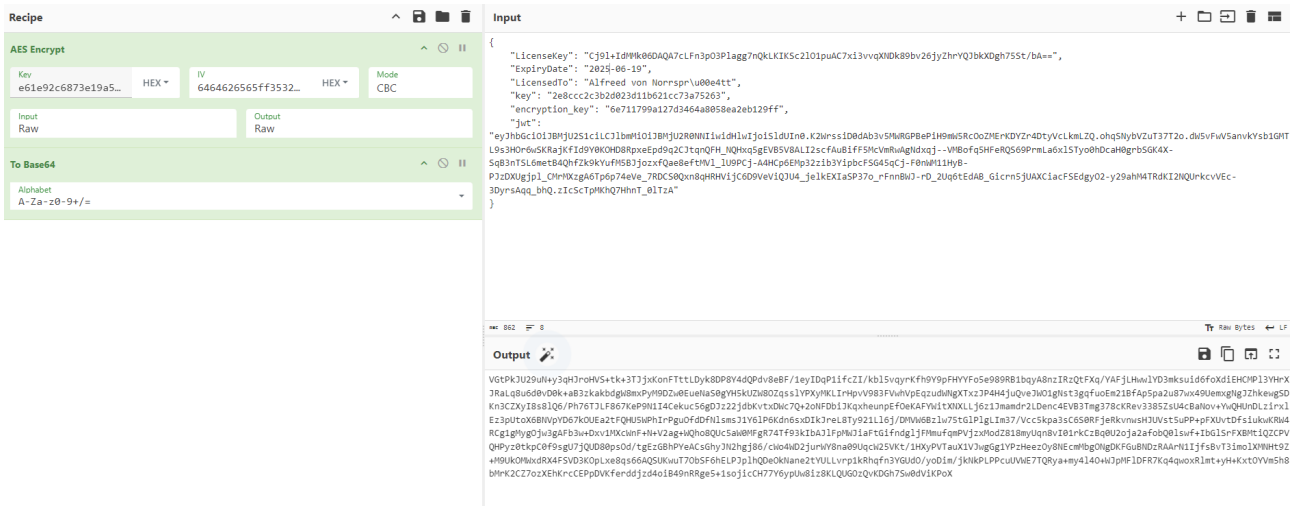
```
// Token: 0x04000003 RID: 3
private static byte[] key = new byte[]
{
    230,
    30,
    146,
    198,
    135,
    62,
    25,
    165,
    9,
    203,
    34,
    109,
    45,
    198,
    236,
    15
};

// Token: 0x04000004 RID: 4
private static byte[] iv = new byte[]
{
    100,
    100,
    98,
    101,
    101,
    byte.MaxValue,
    53,
    50,
    49,
    97,
    54,
    102,
    48,
    99,
    51,
    42
};
```

Vi ser också i klassen att det handlar om AES CBC.

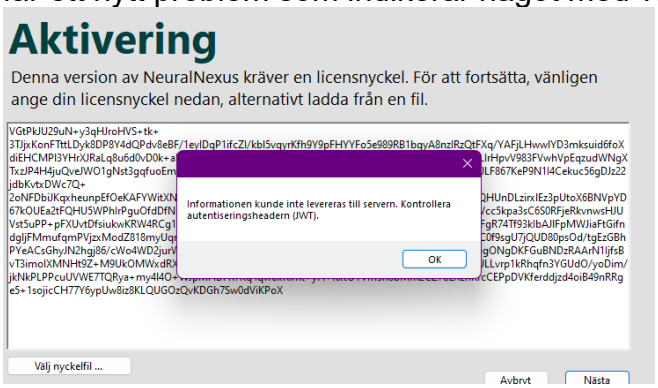
Trycker vi in dehär receptet i Cyberchef tillsammans med våra hemligheter får vi ett json-objekt innehållande licensinformation:



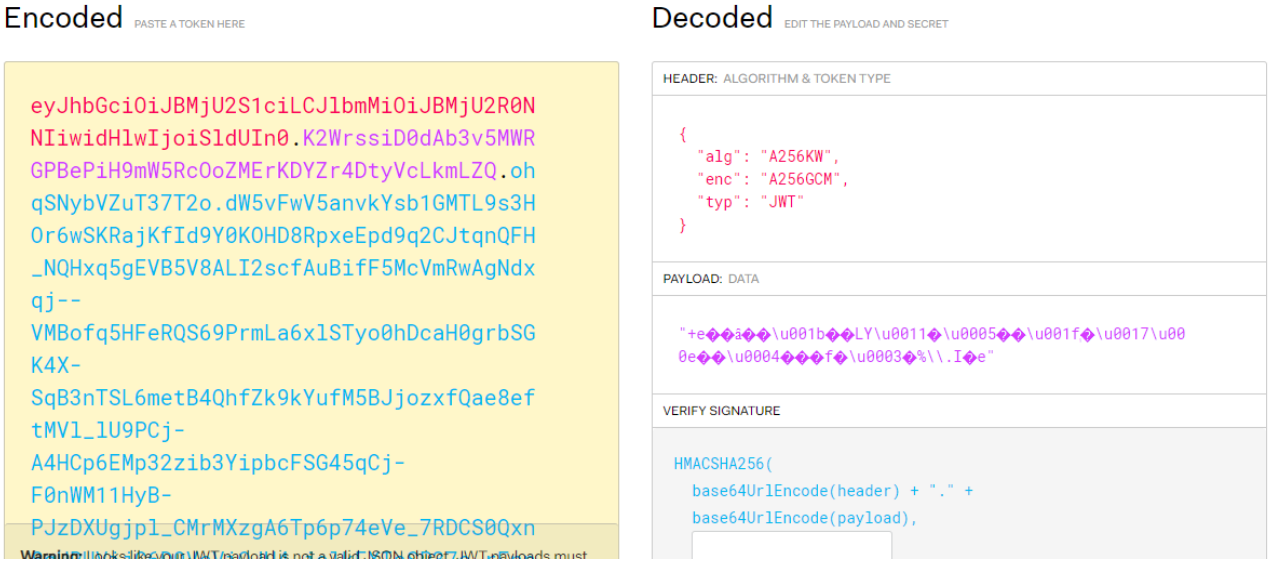
Använder vi samma nyckel för att kryptera om licensdatan med en expirydate nästa år kan vi använda den licensen:



Detta fungerar men vi får ett nytt problem som indikerar något med vår JWT-token:



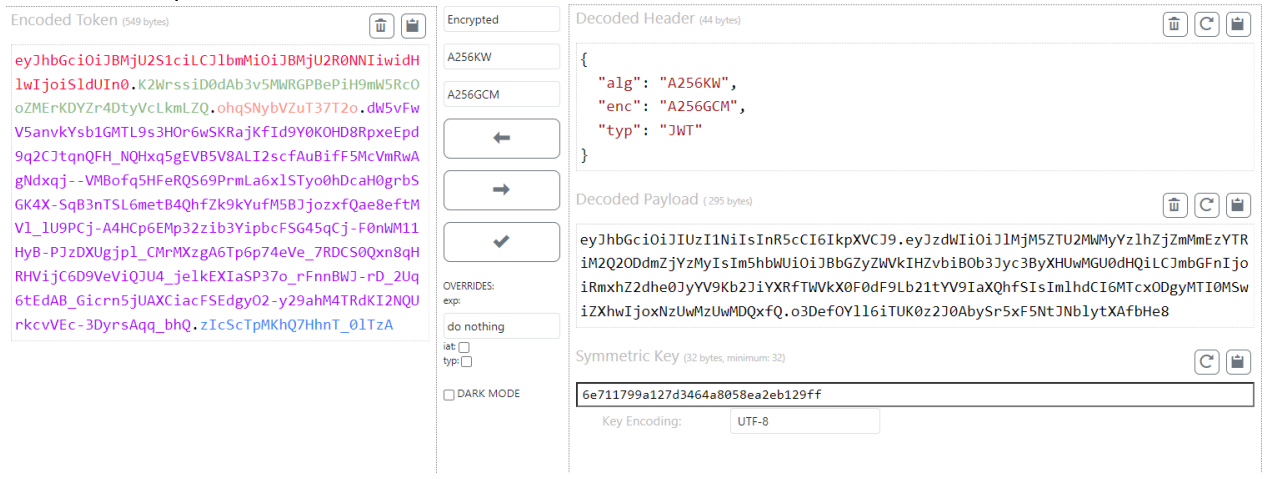
Försöker vi avkoda den JWT med t.ex jwt.io så får vi massa oläsbar data i payloaden. Dock indikerar headern att detta är en krypterad JWT token:



Om vi går tillbaka till vår licensfil så såg vi att det fanns en “key” och “encryption_key” där.


```
{
    "LicenseKey": "Cj9l+IdMMk06DAQA7cLFn3pO3Plagg7nQkLKIK5c2lO1puAC7xi3vvqXNDk89bv26jyZhrYQJbkXDgh75St/bA==",
    "ExpiryDate": "2025-06-19",
    "LicensedTo": "Alfreed von Norrspr\u00e4tt",
    "key": "2e8ccc2c3b2d023d11b621cc73a75263",
    "encryption_key": "6e711799a127d3464a8058ea2eb129ff",
    "jwt":
```


Testar man använda en jwe dekryptor som klarar av krypterade tokens (t.ex https://dinochiesa.github.io/jwt/) så får man en avkrypterad payload (som också verkar vara en JWT)



Kör man sedan denna JWT igenom vilken JWT avkodare som helst så får vi ut flaggan ur payloaden.