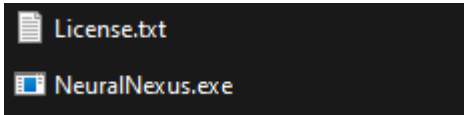


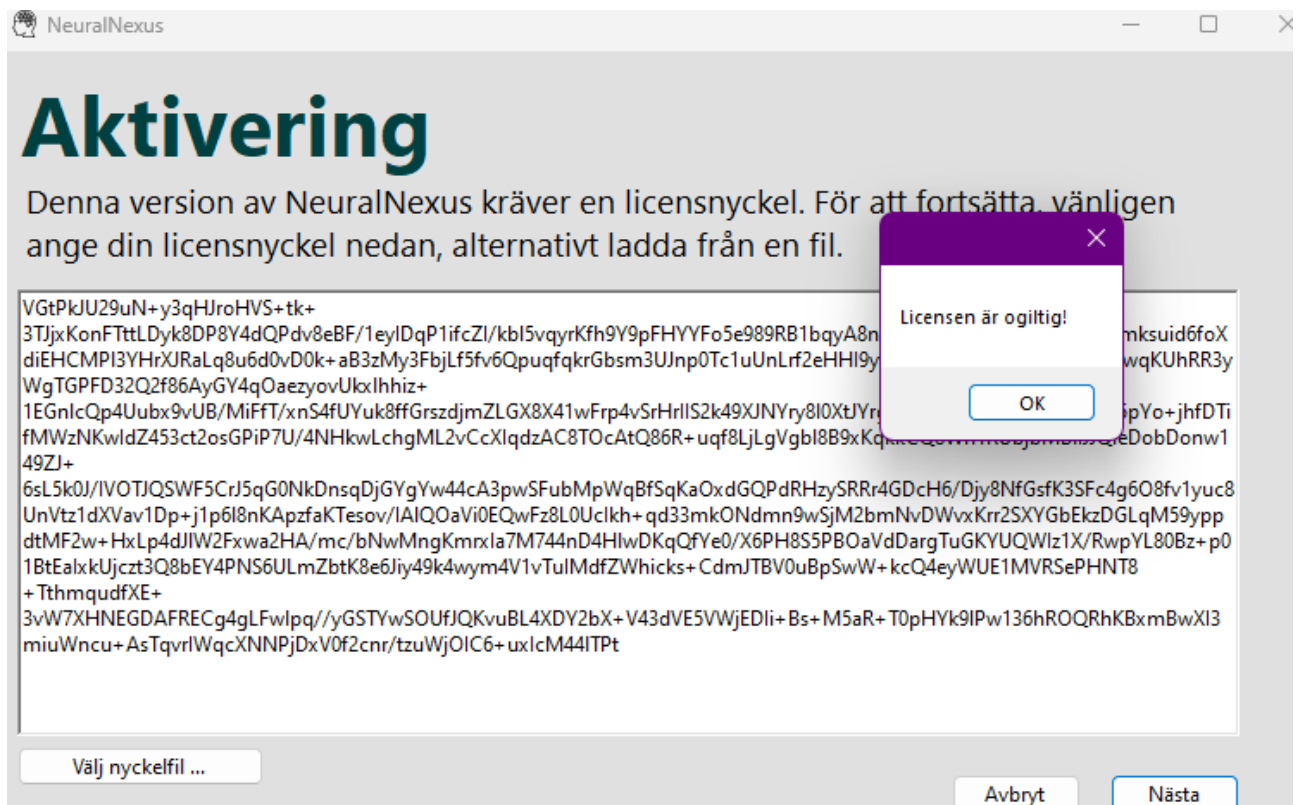
# NeuralNexus

av Challeballe

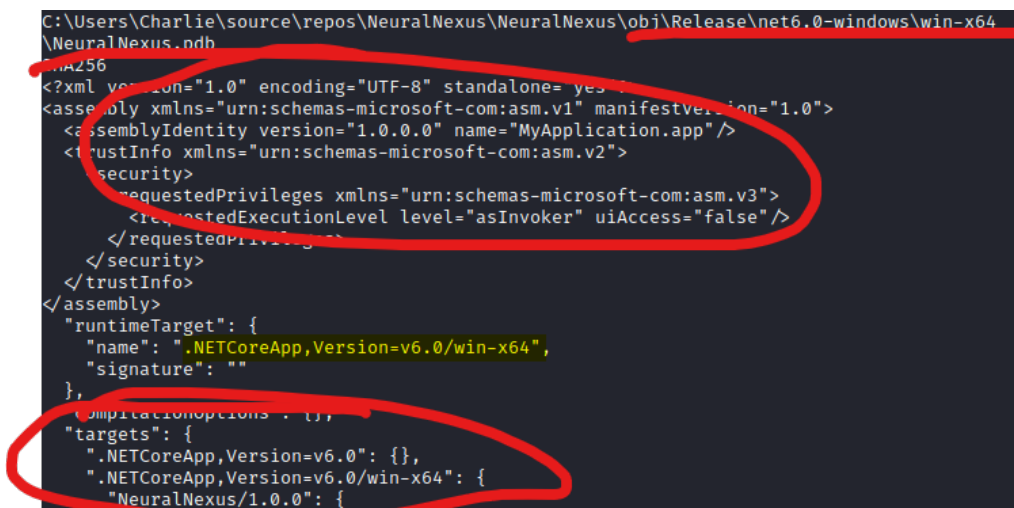
Du möts av två filer, en binär och en txt fil.



Startar man programmet 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:

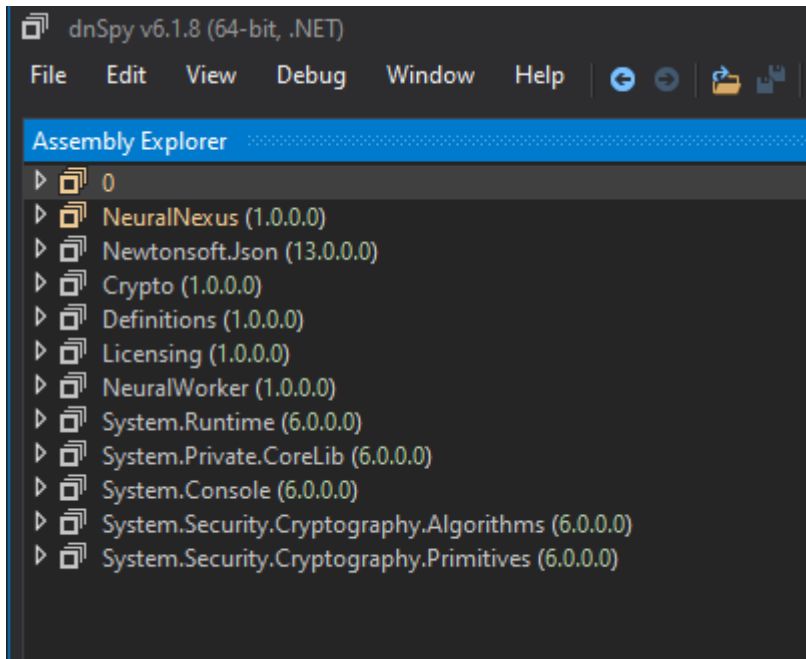
```
{
  "Crypto/1.0.0": {
    "runtime": {
      "Crypto.dll": {}
    }
  },
  "Definitions/1.0.0": {
    "runtime": {
      "Definitions.dll": {}
    }
  },
  "Licensing/1.0.0": {
    "dependencies": {
      "Crypto": "1.0.0",
      "Newtonsoft.Json": "13.0.3"
    },
    "runtime": {
      "Licensing.dll":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

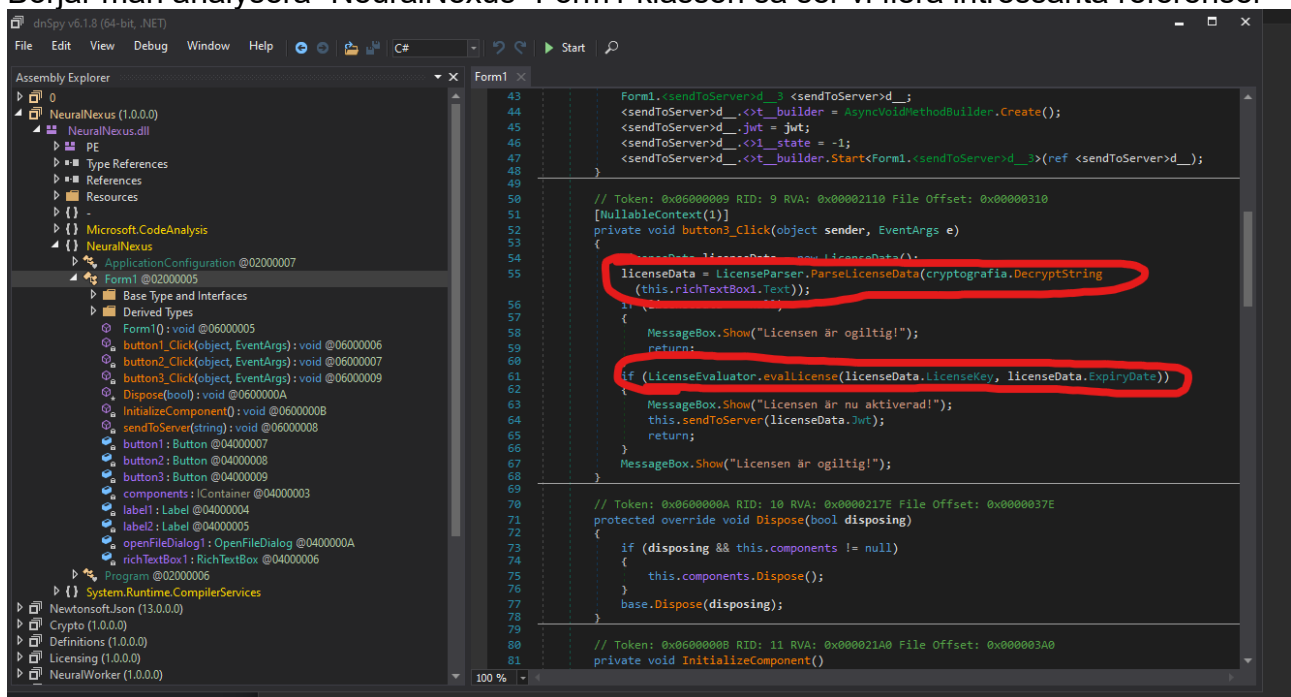
```
$ binwalk --dd='exe' NeuralNexus.exe
```

| 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, sequence length: 12031) |
| 973541  | 0xEDAE5     | Certificate in DER format (x509 v3), header length: 4, sequence length: 1380   |
| 974925  | 0xEE04D     | Certificate in DER format (x509 v3), header length: 4, sequence length: 1680   |
| 976609  | 0xEE6E1     | Certificate in DER format (x509 v3), header length: 4, sequence length: 2015   |
| 979505  | 0xEF231     | Object signature in DER format (PKCS header length: 4, sequence length: 5926)  |
| 979674  | 0xEF2DA     | Certificate in DER format (x509 v3), header length: 4, sequence length: 1728   |
| 981406  | 0xEF99E     | Certificate in DER format (x509 v3), header length: 4, sequence length: 1710   |
| 983120  | 0xF0050     | Certificate in DER format (x509 v3), header length: 4, sequence 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 giltig:

```
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) ==
            "10c8e5217f0128dedb2c843fa82e933d4bbcc0cc5d513c30071ac496996249947";
    }
}
```

Så länge datumet inte är tidigare än just nu och hashen matchar så verkar det tolkas som en giltig 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;
}
```

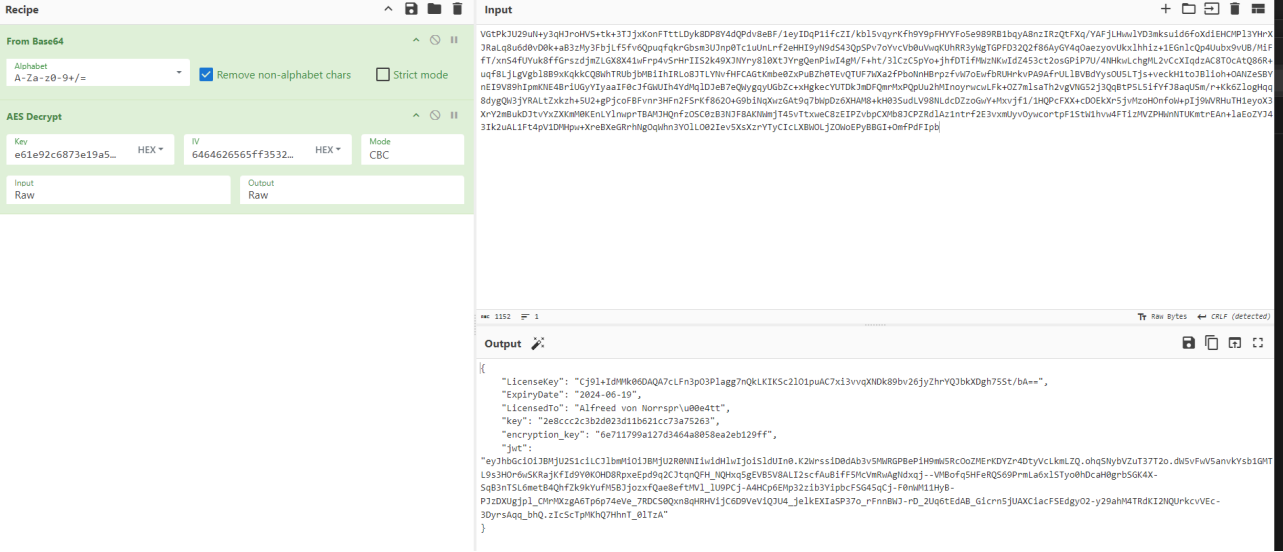
Den kanske mest intressanta klassen i dethär fallet är "cryptografia" som har funktioner för att avkryptera licensfilen. Längst ner i den klassen hittar vi både nyckeln och IVn som krävs för att avkryptera licensfilen:

```
// 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
};

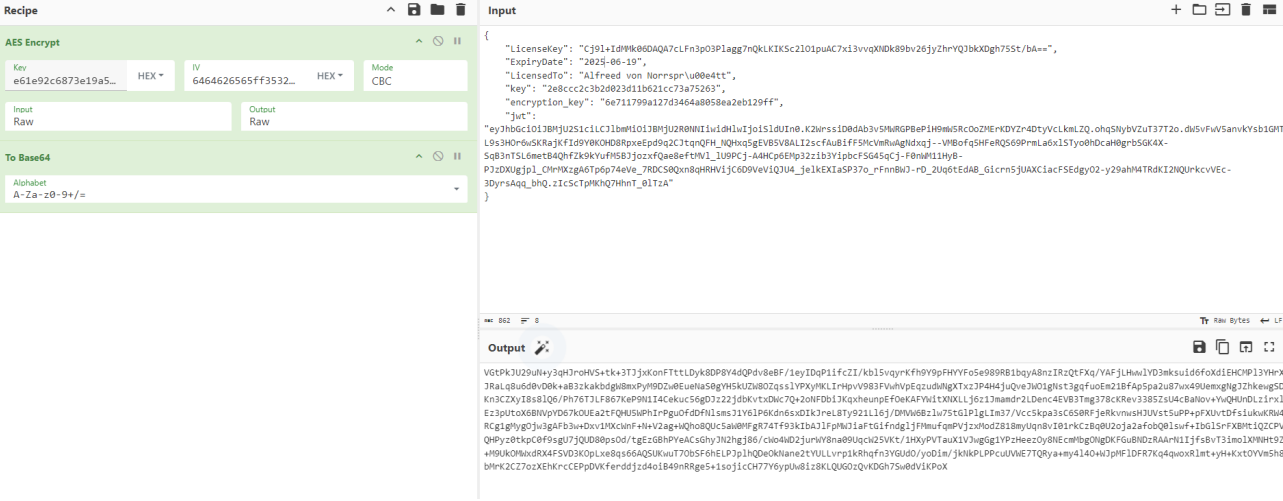
// Token: 0x04000004 RID: 4
private static byte[] iv = new byte[]
{
    100,
    100,
    98,
    101,
    101,
    byte.MaxValue,
    53,
    50,
    49,
    97,
    54,
    102,
    48,
    99,
    51,
    42
};
```

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

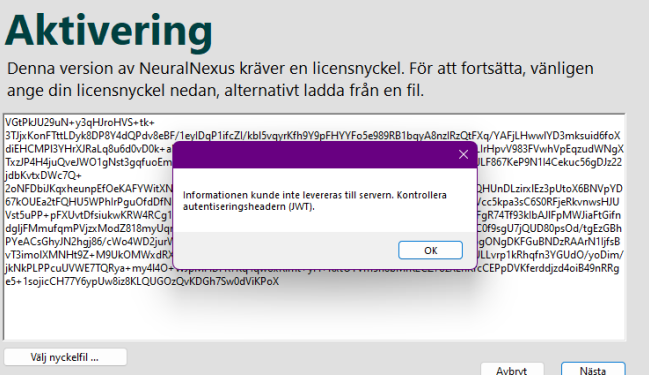
Trycker vi in de här receptet i Cyberchef tillsammans med våra hemligheter får vi ett json-objekt innehållande licensinformation:



Använder vi samma nyckel för att kryptera om licensdata 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:

| Encoded  | Decoded  |
|--|--|
| <div>PASTE A TOKEN HERE</div> <div>eyJhbGciOiJIbWJ2S1ciLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiS1duIn0.K2WrssiD0dAb3v5MWRGPBePiH9mW5Rc0oZMERKDYzr4DtyVcLkmLZQ.ohqSNybVZuT37T2o.dW5vFwV5anvkYsb1GMTL9s3HOr6wSKRajKfId9Y0KOHd8RpxeEpd9q2CJtqnQFH_NQHxq5gEVB5V8ALI2scfAuBiff5McVmRwAgNdxqj--VMBofq5HFeRQS69PrmLa6x1STyo0hDcaH0grbSGK4X-SqB3nTSL6metB4QhfZk9kYufM5BJjozxfQae8eftMV1_lU9PCj-A4HCp6EMp32zib3YipbcFSG45qCj-F0nWM11HyB-PJzDXUgjp1_CMrMXzgA6Tp6p74eVe_7RDCS0Qxn</div> <div>Warning: Links like this JWT Token are not a valid token. The JWT token must</div> | <div>EDIT THE PAYLOAD AND SECRET</div> <div>HEADER: ALGORITHM &amp; TOKEN TYPE</div> <div><pre>{   "alg": "A256KW",   "enc": "A256GCM",   "typ": "JWT" }</pre></div> <div>PAYLOAD: DATA</div> <div><pre>"e\u001b\u0011\u0005\u001f\u0017\u000e\u0004\u000f\u0003%\\".Ie"</pre></div> <div>VERIFY SIGNATURE</div> <div><pre>HMACSHA256(   base64UrlEncode(header) + "." +   base64UrlEncode(payload),</pre></div> |

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

```
{
  "LicenseKey": "Cj9l+IdMMk06DAQ7cLFn3p03Plagg7nQkLKIKSc2l01puAC7xi3vvqXNDk89bv26jyZhrYQJ3bkXDgh75St/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)

| Encoded Token (549 bytes)  | Encrypted  | Decoded Header (44 bytes)   |
|--|--|---|
| <div>eyJhbGciOiJIbWJ2S1ciLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiS1duIn0.K2WrssiD0dAb3v5MWRGPBePiH9mW5Rc0oZMERKDYzr4DtyVcLkmLZQ.ohqSNybVZuT37T2o.dW5vFwV5anvkYsb1GMTL9s3HOr6wSKRajKfId9Y0KOHd8RpxeEpd9q2CJtqnQFH_NQHxq5gEVB5V8ALI2scfAuBiff5McVmRwAgNdxqj--VMBofq5HFeRQS69PrmLa6x1STyo0hDcaH0grbSGK4X-SqB3nTSL6metB4QhfZk9kYufM5BJjozxfQae8eftMV1_lU9PCj-A4HCp6EMp32zib3YipbcFSG45qCj-F0nWM11HyB-PJzDXUgjp1_CMrMXzgA6Tp6p74eVe_7RDCS0Qxn8qHRHVjC6D9veViqJU4_je1kEXIaSP37o_rFnnBWJ-rD_2Uq6tEdAB_Gicrn5jUAXCIacFSEdyO2-y29ahM4TRdKI2NQURkcvVEc-3DyrsAqq_bhQ.zIcScTpMKhQ7Hhnt_01TzA</div> | <div>A256KW</div> <div>A256GCM</div> <div>←</div> <div>→</div> <div>✓</div> <div>OVERRIDES:<br/>exp:<br/>iat:<br/>typ:</div> <div><input type="checkbox"/> DARK MODE</div> | <div>{<br/>  "alg": "A256KW",<br/>  "enc": "A256GCM",<br/>  "typ": "JWT"<br/>}</div> <div>Decoded Payload (295 bytes)</div> <div>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJlMjU2R0NNIiwidHlwIjoiS1duIn0.K2WrssiD0dAb3v5MWRGPBePiH9mW5Rc0oZMERKDYzr4DtyVcLkmLZQ.ohqSNybVZuT37T2o.dW5vFwV5anvkYsb1GMTL9s3HOr6wSKRajKfId9Y0KOHd8RpxeEpd9q2CJtqnQFH_NQHxq5gEVB5V8ALI2scfAuBiff5McVmRwAgNdxqj--VMBofq5HFeRQS69PrmLa6x1STyo0hDcaH0grbSGK4X-SqB3nTSL6metB4QhfZk9kYufM5BJjozxfQae8eftMV1_lU9PCj-A4HCp6EMp32zib3YipbcFSG45qCj-F0nWM11HyB-PJzDXUgjp1_CMrMXzgA6Tp6p74eVe_7RDCS0Qxn8qHRHVjC6D9veViqJU4_je1kEXIaSP37o_rFnnBWJ-rD_2Uq6tEdAB_Gicrn5jUAXCIacFSEdyO2-y29ahM4TRdKI2NQURkcvVEc-3DyrsAqq_bhQ.zIcScTpMKhQ7Hhnt_01TzA</div> <div>Symmetric Key (32 bytes, minimum: 32)</div> <div>6e711799a127d3464a8058ea2eb129ff</div> <div>Key Encoding: UTF-8</div> |

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