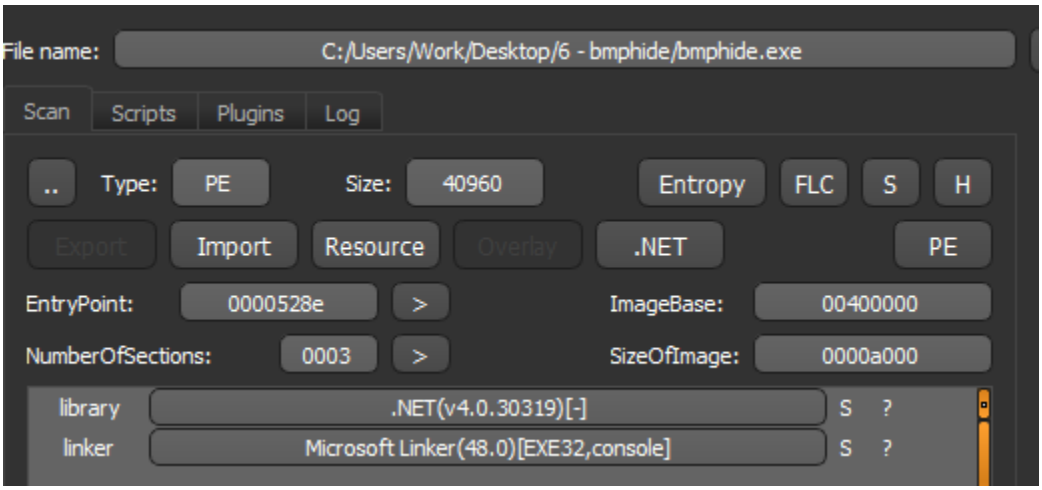
Bmphide

Check chương trình ta thấy chương trình được viết bằng C#



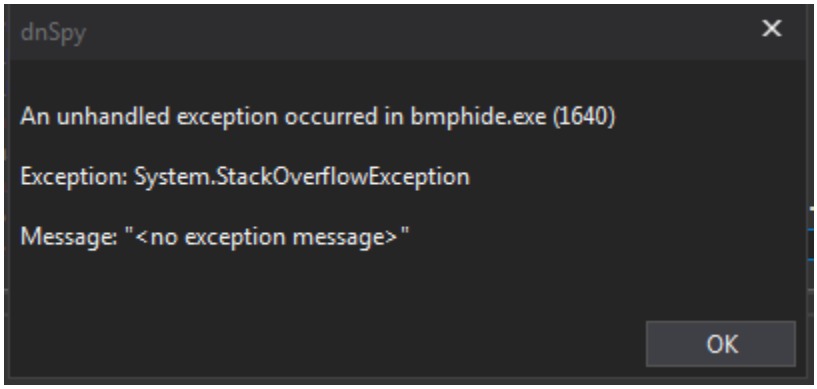
Hàm Main của chương trình khi mở bằng dnSpy

```
private static void Main(string[] args)
{
    Program.Init();
    Program.yy += 18;
    string filename = args[2];
    string fullPath = Path.GetFullPath(args[0]);
    string fullPath2 = Path.GetFullPath(args[1]);
    byte[] data = File.ReadAllBytes(fullPath2);
    Bitmap bitmap = new Bitmap(fullPath);
    byte[] data2 = Program.h(data);
    Program.i(bitmap, data2);
    bitmap.Save(filename);
}
```

Dựa theo hàm main ta có thể thấy chương trình này có 3 input, 1 là ảnh gốc dùng để giấu dữ liệu encode, 2 là dữ liệu muốn encode, 3 là file ảnh đã encode. Cách thực hoạt động như sau:

- Tạo một ảnh mới giống input 1 trong memory
- Encode dữ liệu input 2 bằng hàm h
- Trộn dữ liệu đã mã hóa vào ảnh trong memory bằng hàm i
- Ghi lại ảnh mới vào ổ cứng

Sau khi debug trên dnSpy thì bị báo lỗi, trong khi chạy bình thường thì không bị



Trong A.IndentifyLocals ta sẽ tìm thấy đoạn code sau

```
A.originalDelegate = (A.locateNativeCallingConvention)Marshal.GetDelegateForFunctionPointer(intPtr2, typeof
  (A.locateNativeCallingConvention));
A.handler = new A.locateNativeCallingConvention(A.IncrementMaxStack);
RuntimeHelpers.PrepareDelegate(A.originalDelegate);
RuntimeHelpers.PrepareDelegate(A.handler);
```

Đoạn code này nhằm che giấu đoạn code thực sự được chạy khi debug bằng dnSpy

Đến chỗ này thì mình không biết cách để lấy được đoạn code thực như thế nào nên mình sử dụng một cách khác để xem cách hoạt động của chương trình là cho input thứ 2 của chương trình là một file thật to để mình có thời gian cho dnSpy attach debug vào chương trình đang chạy lúc chương trình đang encode dữ liệu

Đây là hàm h trong dnSpy

```
public static byte[] h(byte[] data)
{
    byte[] array = new byte[data.Length];
    int num = 0;
    for (int i = 0; i < data.Length; i++)
    {
        int num2 = (int)Program.f(num++);
        int num3 = (int)data[i];
        num3 = (int)Program.e((byte)num3, (byte)num2);
        num3 = (int)Program.a((byte)num3, 7);
        int num4 = (int)Program.f(num++);
        num3 = (int)Program.e((byte)num3, (byte)num4);
        num3 = (int)Program.c((byte)num3, 3);
        array[i] = (byte)num3;
    }
    return array;
}
```

Sau khi step in hàm f thì phát hiện nó không chạy hàm f mà chạy hàm g

```
// Token: 0x06000014 RID: 20 RVA: 0x000028F0 File Offset: 0x000028F0
public static byte g(int idx)
{
    byte b = (byte)((long)(idx + 1) * (long)((ulong)-306674912));
    byte k = (byte)((idx + 2) * 1669101435);
    return Program.e(b, k);
}
```

Tuy nhiên sau khi code hàm g bằng visual studio và chạy thử thì không ra đúng kết quả trả về khi debug, nên tới đây vẫn chưa biết được hàm chạy thật sự ở chỗ này là gì

Kế tiếp, do không biết code để trả kết quả là gì, mình in ra một mảng các giá trị trả về của hàm này và tìm được quy luật của hàm là cứ sau 128 lần chạy thì sẽ chạy lại giá trị ban đầu, mảng này có 128 phần tử có giá trị

```
int[] mask = new int[128]
            {
                63, 253, 187, 101, 55, 245, 139, 77,
                15, 237, 171, 101, 215, 149, 91, 29,
                223, 157, 91, 37, 151, 213, 11, 77,
                143, 45, 107, 165, 247, 53, 123, 189,
                255, 61, 123, 165, 247, 53, 139, 205,
                15, 173, 107, 37, 215, 149, 91, 29,
                223, 157, 91, 229, 151, 85, 11, 205,
                143, 109, 43, 229, 183, 117, 59, 253,
                63, 125, 187, 229, 55, 117, 139, 205,
```

```
                15, 109, 171, 229, 87, 149, 219, 29,
                95, 157, 219, 37, 151, 85, 11, 205,
                143, 45, 235, 165, 119, 53, 251, 189,
                127, 61, 251, 165, 119, 53, 139, 77,
                15, 173, 235, 37, 87, 149, 219, 29,
                95, 157, 219, 101, 151, 213, 11, 77,
                143, 237, 43, 101, 183, 245, 59, 125

            };
```

Tiếp theo debug những hàm tiếp theo thì ta thấy hàm h thay hàm a bằng b và c bằng d, hàm h sẽ chạy như sau:

```
for (int i = 0; i < data.Length; i++)
       {
                    int num2 = (int)Program.f(num++);
                    int num3 = (int)data[i];
                    num3 = (int)Program.e((byte)num3, (byte)num2);
                    num3 = (int)Program.b((byte)num3, 7);
                    int num4 = (int)Program.f(num++);
                    num3 = (int)Program.e((byte)num3, (byte)num4);
                    num3 = (int)Program.d((byte)num3, 3);
                    array[i] = (byte)num3;
       }
return array;
```

Từ đây ta có thể viết hàm bruteforce để tìm được data ban đầu của input 2:

```
public static void h(byte[] data, byte[] encoded_data)
        {
            int[] mask = new int[128]
            {
                63, 253, 187, 101, 55, 245, 139, 77,
                15, 237, 171, 101, 215, 149, 91, 29,
                223, 157, 91, 37, 151, 213, 11, 77,
                143, 45, 107, 165, 247, 53, 123, 189,
                255, 61, 123, 165, 247, 53, 139, 205,
                15, 173, 107, 37, 215, 149, 91, 29,
                223, 157, 91, 229, 151, 85, 11, 205,
                143, 109, 43, 229, 183, 117, 59, 253,
                63, 125, 187, 229, 55, 117, 139, 205,
                15, 109, 171, 229, 87, 149, 219, 29,
                95, 157, 219, 37, 151, 85, 11, 205,
                143, 45, 235, 165, 119, 53, 251, 189,
                127, 61, 251, 165, 119, 53, 139, 77,
                15, 173, 235, 37, 87, 149, 219, 29,
                95, 157, 219, 101, 151, 213, 11, 77,
                143, 237, 43, 101, 183, 245, 59, 125
            };
            int num = 0;
            int temp = 0;
            for (int i = 0; i < data.Length; i++)
            {

                temp = num;
```

```
                num %= 128;
                int num2 = mask[num];
                num++;
                int num3 = (int)data[i];
                num3 = (int)Program.e((byte)num3, (byte)num2);
                num3 = (int)Program.b((byte)num3, 7);
                num %= 128;
                int num4 = mask[num];
                num++;
                num3 = (int)Program.e((byte)num3, (byte)num4);
                num3 = (int)Program.d((byte)num3, 3);
                if((byte)num3 != encoded_data[i])
                {
                    num = temp;
                    data[i] += 1;
                    i--;
                    continue;
                }
            }
        }
```

Tiếp theo ta phân tích hàm i để xem chương trình nhúng data đã mã hóa vào ảnh như thế nào

```
int num = Program.j(103);
for (int i = Program.j(103); i < bm.Width; i++)
{
    for (int j = Program.j(103); j < bm.Height; j++)
    {
        bool flag = num > data.Length - Program.j(231);
        if (flag)
        {
            break;
        }
        Color pixel = bm.GetPixel(i, j);
        int red = ((int)pixel.R & Program.j(27)) | ((int)data[num] & Program.j(228));
        int green = ((int)pixel.G & Program.j(27)) | (data[num] >> Program.j(230) & Program.j(228));
        int blue = ((int)pixel.B & Program.j(25)) | (data[num] >> Program.j(100) & Program.j(230));
        Color color = Color.FromArgb(Program.j(103), red, green, blue);
        bm.SetPixel(i, j, color);
        num += Program.j(231);
    }
}
```

Sau khi debug thì ta có

```
j(25) = 252 - 1111 1100
j(27) = 248 - 1111 1000
j(100) = 0x6 - 0000 0110
j(103) = 0x0 - 0000 0000
j(228) = 0x7 - 0000 0111
j(230) = 0x3 - 0000 0011
j(231) = 0x1 - 0000 0001
```

Từ đó ta có hàm chạy như sau:

```
int red = ((int)pixel.R & 0b11111000) | ((int)data[num] & 0b0111);
int green = ((int)pixel.G & 0b11111000) | (data[num] >> 3 & 0b0111);
int blue = ((int)pixel.B & 0b11111100) | (data[num] >> 6 & 0b0011);
Color color = Color.FromArgb(0, red, green, blue);
```

Pixel có 3 kênh màu là red green blue, mỗi byte của dữ liệu mã hóa sẽ được giấu vào 3 kênh màu của mỗi pixel ảnh, 3 bits đầu tiên của byte dữ liệu giấu vào 3 right-most-bits của kênh red, tương tự 3 bits tiếp theo giấu vào green, và 2 bits cuối giấu vào màu blue

Từ đó ta có hàm lấy dữ liệu như sau:

```
public static void i(Bitmap bm, byte[] data)
        {
            int num = 0;
            for (int i = 0; i < bm.Width; i++)
            {
                for (int j = 0; j < bm.Height; j++)
                {
                    bool flag = num > data.Length - 1;
                    if (flag)
                    {
                        break;
                    }
                    Color pixel = bm.GetPixel(i, j);
                    data[num] = (byte)0;
                    data[num] = (byte)((int)data[num] | (pixel.B & 3));
                    data[num] = (byte)(((int)data[num] << 3) | (pixel.G & 7));
                    data[num] = (byte)(((int)data[num] << 3) | (pixel.R & 7));
                    num += 1;
                }
            }

        }
```

Tiếp theo là hàm main:

```
static void Main(string[] args)
        {
            string filename = args[1];
            string fullPath = Path.GetFullPath(args[0]);
            Bitmap bitmap = new Bitmap(fullPath);
            byte[] data2 = new byte[(bitmap.Height * bitmap.Width)];
            Program.i(bitmap, data2);
            byte[] data1 = new byte[bitmap.Height * bitmap.Width];
            for (int i = 0; i < data1.Length; i++)
                data1[i] = 0;
            h(data1, data2);
            using (FileStream stream = new FileStream(filename, FileMode.Create))
```

```
            {
                using (BinaryWriter writer = new BinaryWriter(stream))
                {
                    writer.Write(data1);
                    writer.Write(data1.Length);
                    writer.Close();

                }
            }

        }
```

Sau khi decode ảnh của challenge ta có 1 ảnh mới là:

Lấy ảnh này decode tiếp thì được: