This document contains concepts and instructions that may be hard to understand. This is **not a test**. This is **not a trick**.

If anything is unclear, stop reading the document then ask these questions:

* Can I clearly say what I **think** I am being asked to do?
* Have I read the entire instruction and do I understand **all** of it?
* Can I tell someone what I think I understand?
* Can I ask someone what they understand about the instruction?

If the answer is “No” to any of these questions, do not proceed further.

Instead, take the document to your co-worker, stream leader or supervisor and work through the instruction with them.

If the document is unclear, it will be updated to make things clear but **we need your feedback!**

**Stop!**

In this document:

Table of Contents

[Pre-requisites 2](#_Toc23234637)

[Reason for this Tutorial 3](#_Toc23234638)

[Expected Outcomes 3](#_Toc23234639)

[What is not covered 3](#_Toc23234640)

[Background 3](#_Toc23234641)

[Install and use a C# Decompiler 4](#_Toc23234642)

[Using ILDASM 4](#_Toc23234643)

[Using Dot Peek 11](#_Toc23234644)

[Online Tools 11](#_Toc23234645)

[Free Tools from the Internet 11](#_Toc23234646)

[Write a Report on the code found 12](#_Toc23234647)

[Hints 15](#_Toc23234648)

# Pre-requisites

The instructions in this document were written for a Windows 10 PC running Chrome or Microsoft Edge browsers. The reader should be reasonably familiar with both the Windows 10 operating system, the Windows Desktop interface and the preferred browser.

The reader should have successfully completed the tutorial [Getting Started with MASM](https://github.com/mnewbery/MASM/blob/master/Dandelion%20Getting%20Started%20with%20MASM%200_4.docx) and successfully submitted a Capture-The-Flag report

The reader should have local administrator permission on the computer being used.

The reader needs to already have a good understanding of how to download and install products from repositories such as [Github](https://github.com/mnewbery/MASM/) and [JetBrains](https://www.jetbrains.com/decompiler/download/#section=standalone).

Where gaps in required knowledge are identified, view the Tools and Techniques section for more background information.

# Reason for this Tutorial

Writing and understanding code assembled with C# or C++ then viewing the both the output of the compiler and the output of the executed code is a good way to learn how to find and understand malware.

This tutorial will guide the reader through steps to install a C# specific de-compiler and use it to extract information for further analysis.

Skills, knowledge and experience gained through this tutorial are expected to be foundation knowledge for anyone wanting to start an exciting career as a Cyber Security Defender, Cyber Threat Emulator or Offensive Cyber Security Professional.

# Expected Outcomes

At the end of this tutorial, the reader will be able to:

1. Understand the differences between a Macro Assembler binary and a binary output from a C# compiler
2. Use different IDEs to demonstrate analysis of a compiled binary
3. Extract data from the binary and use this to create a report
4. View and understand the output of the compiled code; and
5. Write a report on the code found using a [**Capture-The-Flag**](https://en.wikipedia.org/wiki/Capture_the_flag#Computer_security) report style

# What is not covered

This tutorial provides a foundation on viewing and understanding byte code that was specifically output from a C# based compiler. It does not cover the various compiler versions and settings that can influence the final byte code output. Code obfuscation and self-modifying code are advanced topics not covered here.

# Background

This tutorial focusses on the 32 bit Complex Instruction Set (CISC) assembly language and compiler normally associated with Windows x86 and x64.

The next section explains how to download and install a typical Integrated Design Environment (IDE) that will decompile a sample C# binary.

# Install and use a C# Decompiler

Two IDEs are provided in this tutorial. The reader should be familiar with both.

**ALWAYS RUN INSTALLERS AS ADMINISTRATOR!** To run an installer as administrator, right click on the installer file then select “Run as administrator” from the menu that appears.

## Using ILDASM

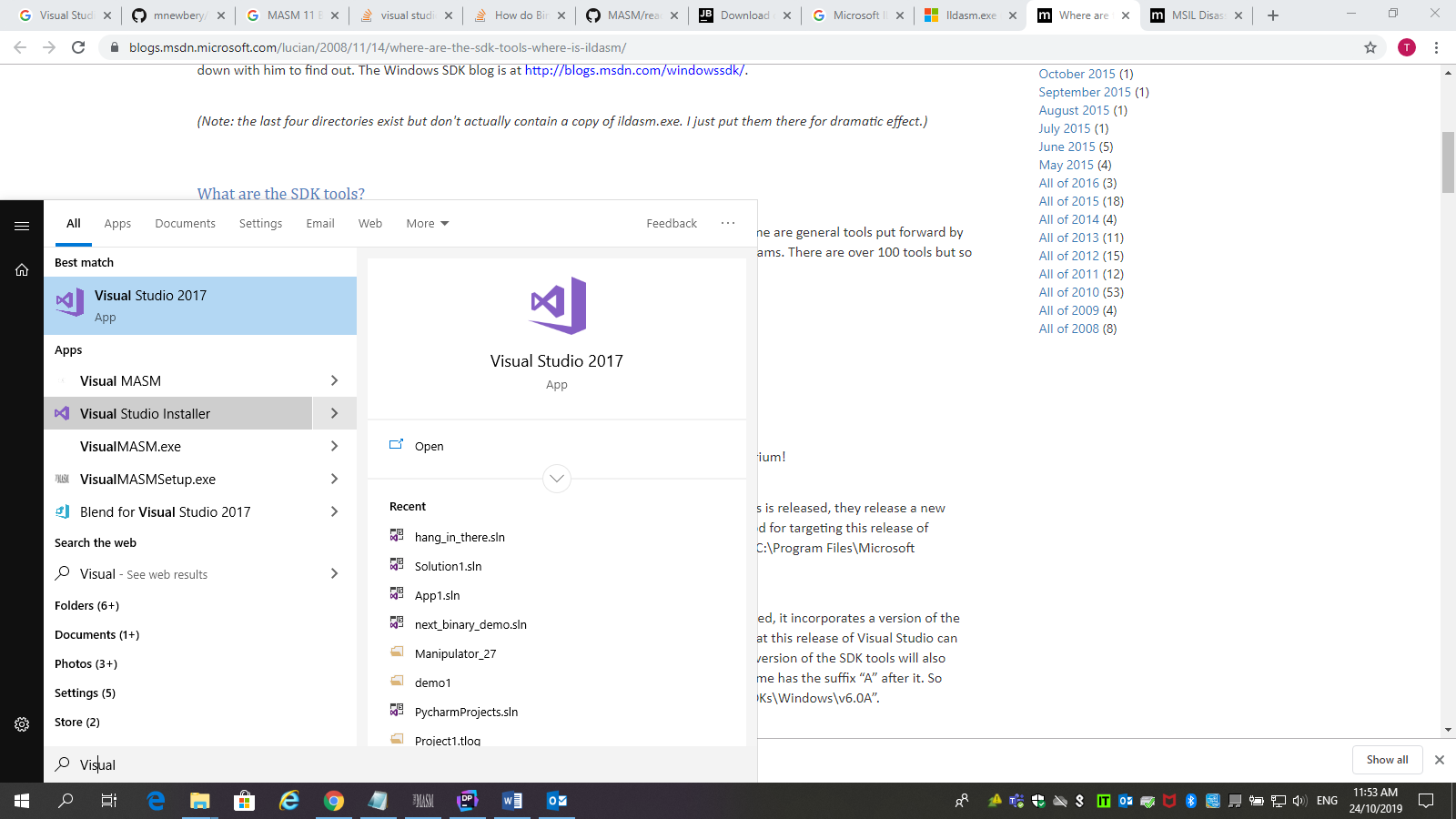
ILDASM is installed as part of Visual Studio. It is not normally downloaded as a stand-alone executable  
  
Instructions for the use of ILDASM.exe appear [here](https://docs.microsoft.com/en-us/dotnet/framework/tools/ildasm-exe-il-disassembler)

The binary for ILDASM.exe (if already installed) will appear in the following location for a 32-bit install of visual studio:

[C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.6.1 Tools](file:///C:\Program%20Files%20(x86)\Microsoft%20SDKs\Windows\v10.0A\bin\NETFX%204.6.1%20Tools)

If the file ILDASM.exe does not appear in this folder or the folder doesn’t exist, follow the steps in the next section to install the tools. Alternatively, skip the next section and use the tool!

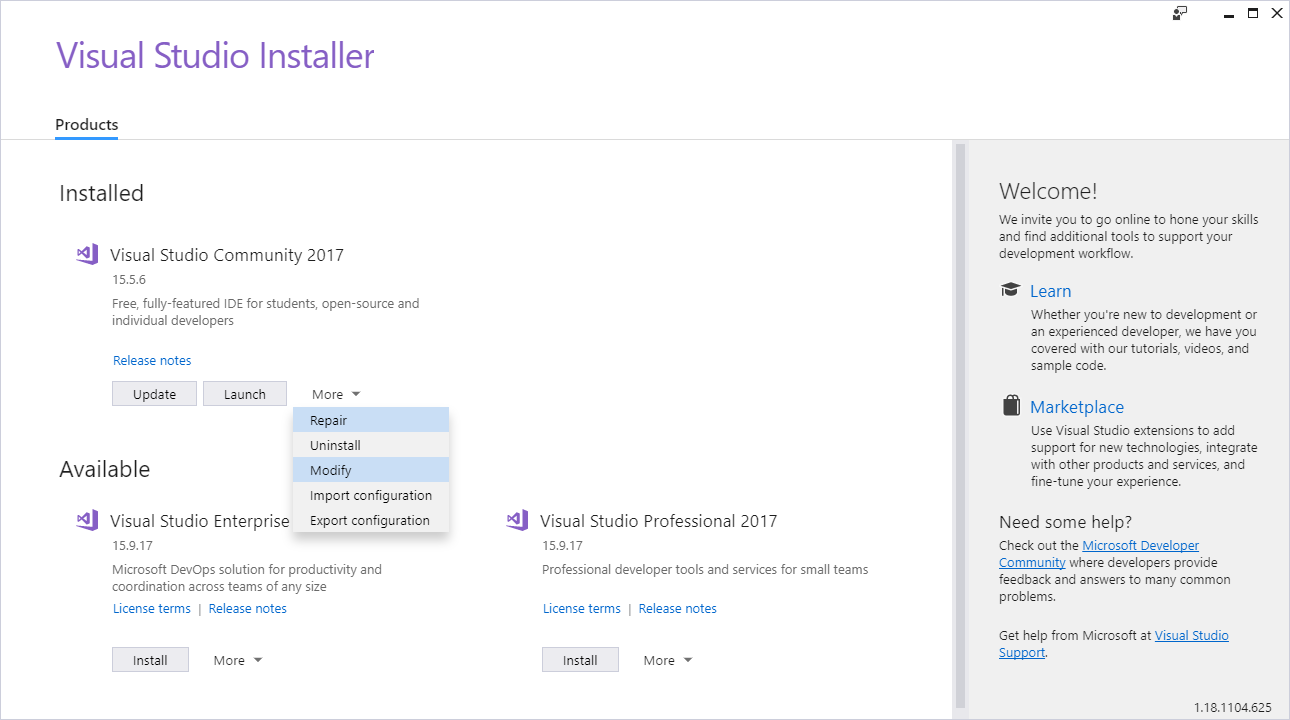
Installation of ILDASM for Visual Studio 2017 and later appears below. Start by stopping Visual studio if it is running then executing Visual Studio *Installer*. See below:



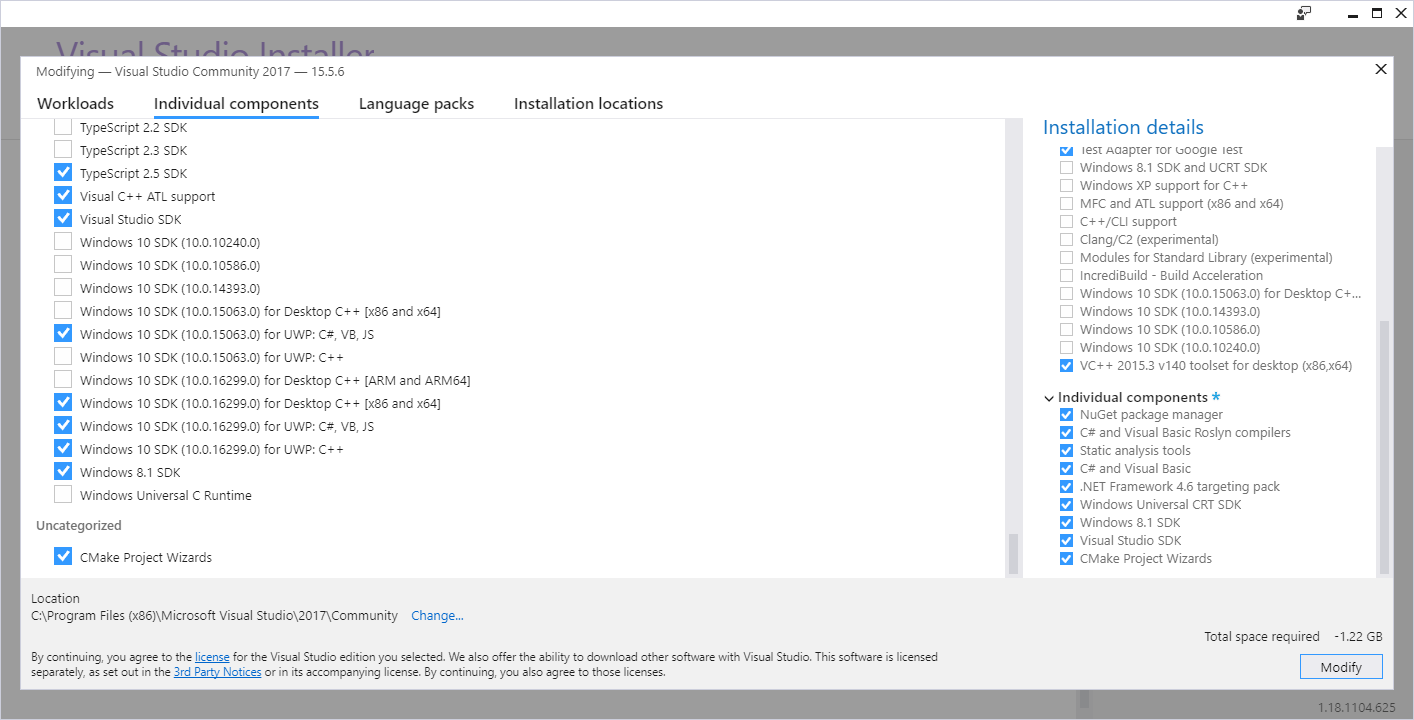
When the confirmation message “Do you want to allow this app to make changes to your device” appears, click Yes to continue.

The installer dialog will appear.

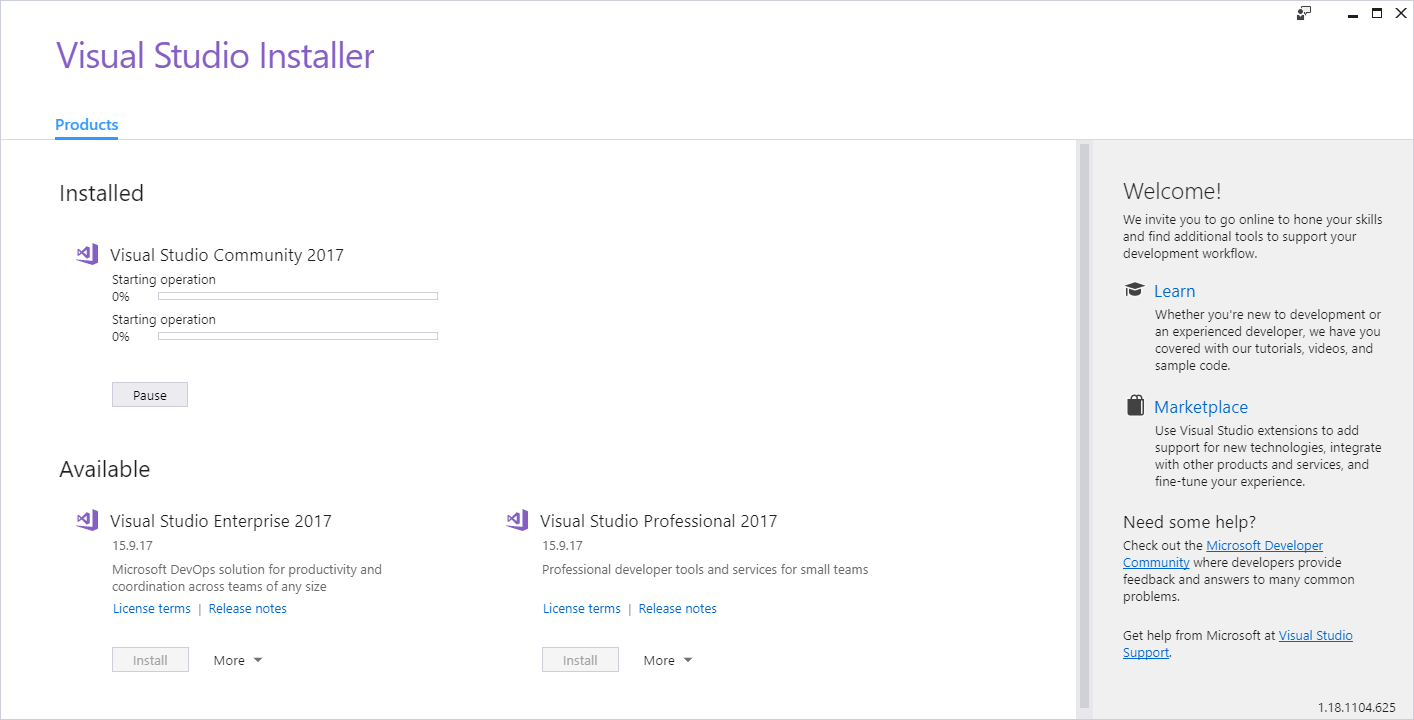
Click the “More” dropdown then choose Modify to continue. See below:



Check the box for Individual Components > Visual Studio SDK then click modify



Wait for the changes to be installed. This could take many minutes. See below



Once the installed has completed, there is no need to launch Visual Studio at this time. The executable ILDASM.exe is the only component required.

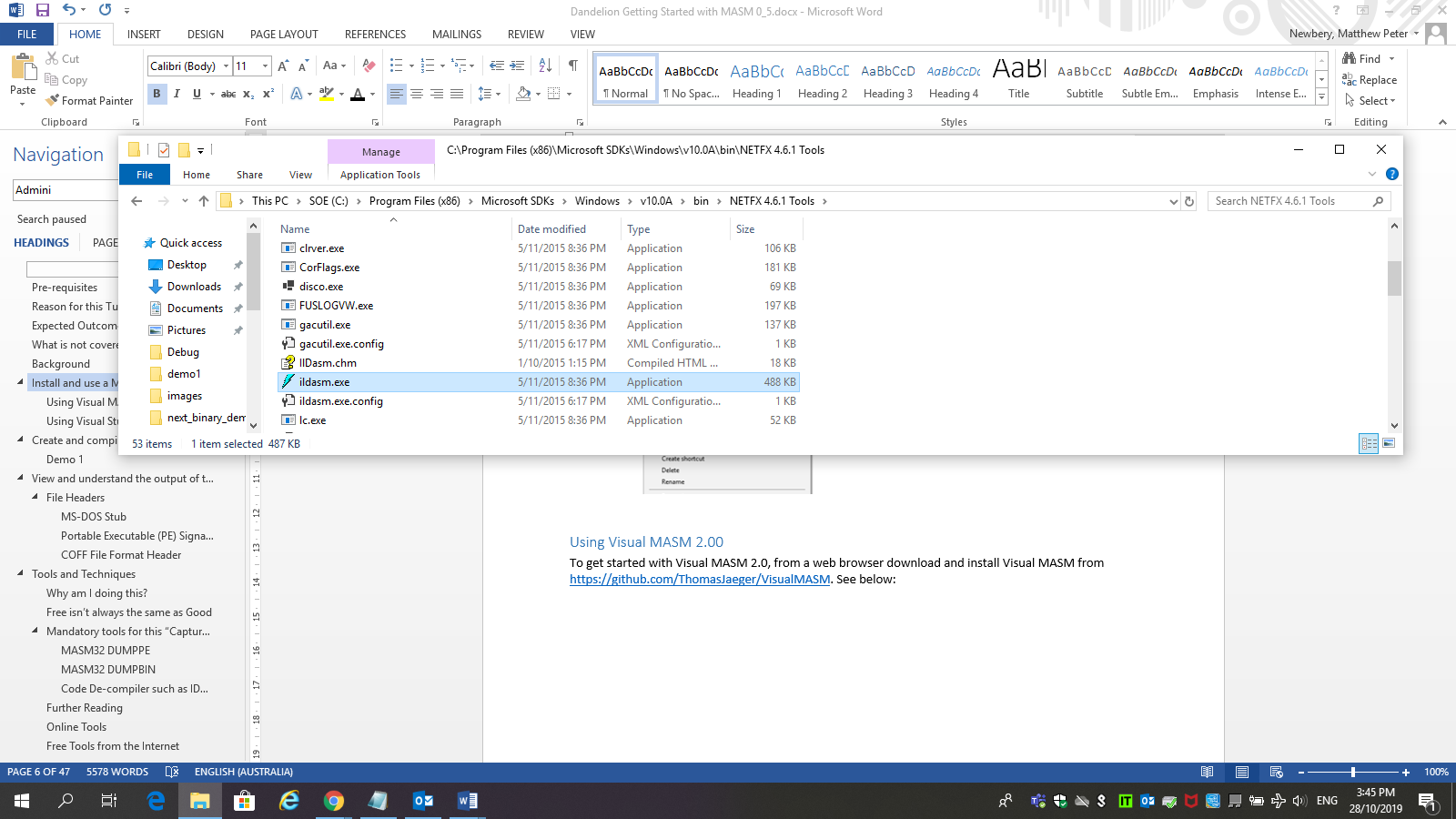
Use ILDASM to view the contents of a binary file

Follow the steps below to view the ILDASM user interface, load a binary then view the content of the binary

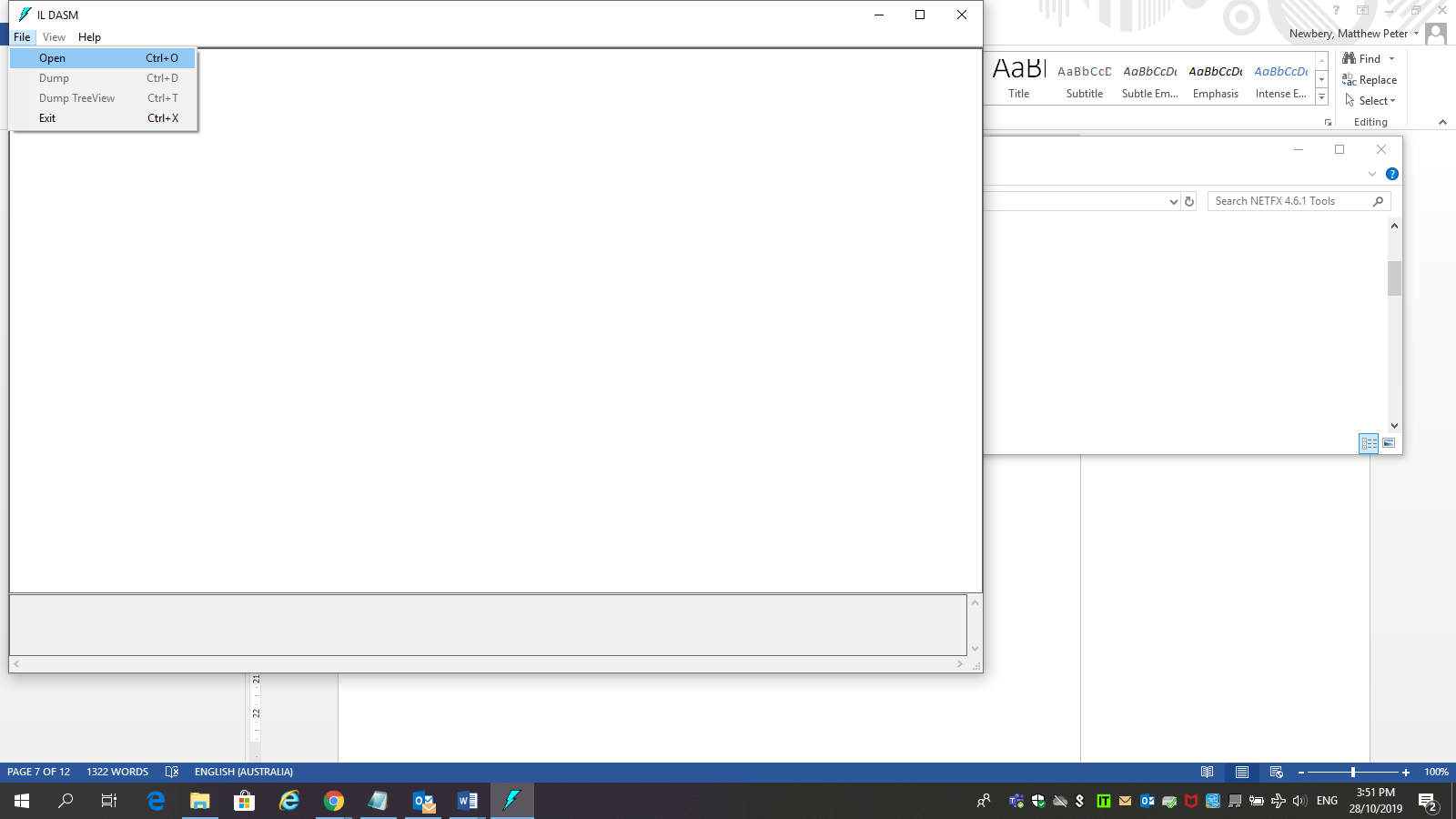
Navigate to ILDASM.exe. This file should be in the following path:

C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.6.1 Tools

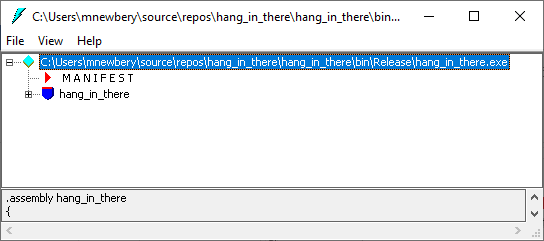
Below is an image of the file in the path:



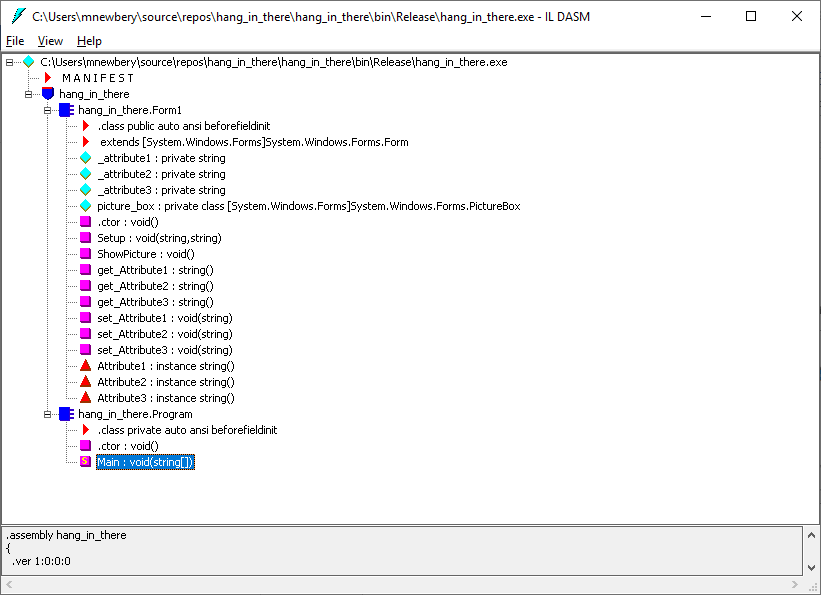
**Run ILDASM as administrator** To run an executable as administrator, right click on the installer file then select “Run as administrator” from the menu that appears. See below:



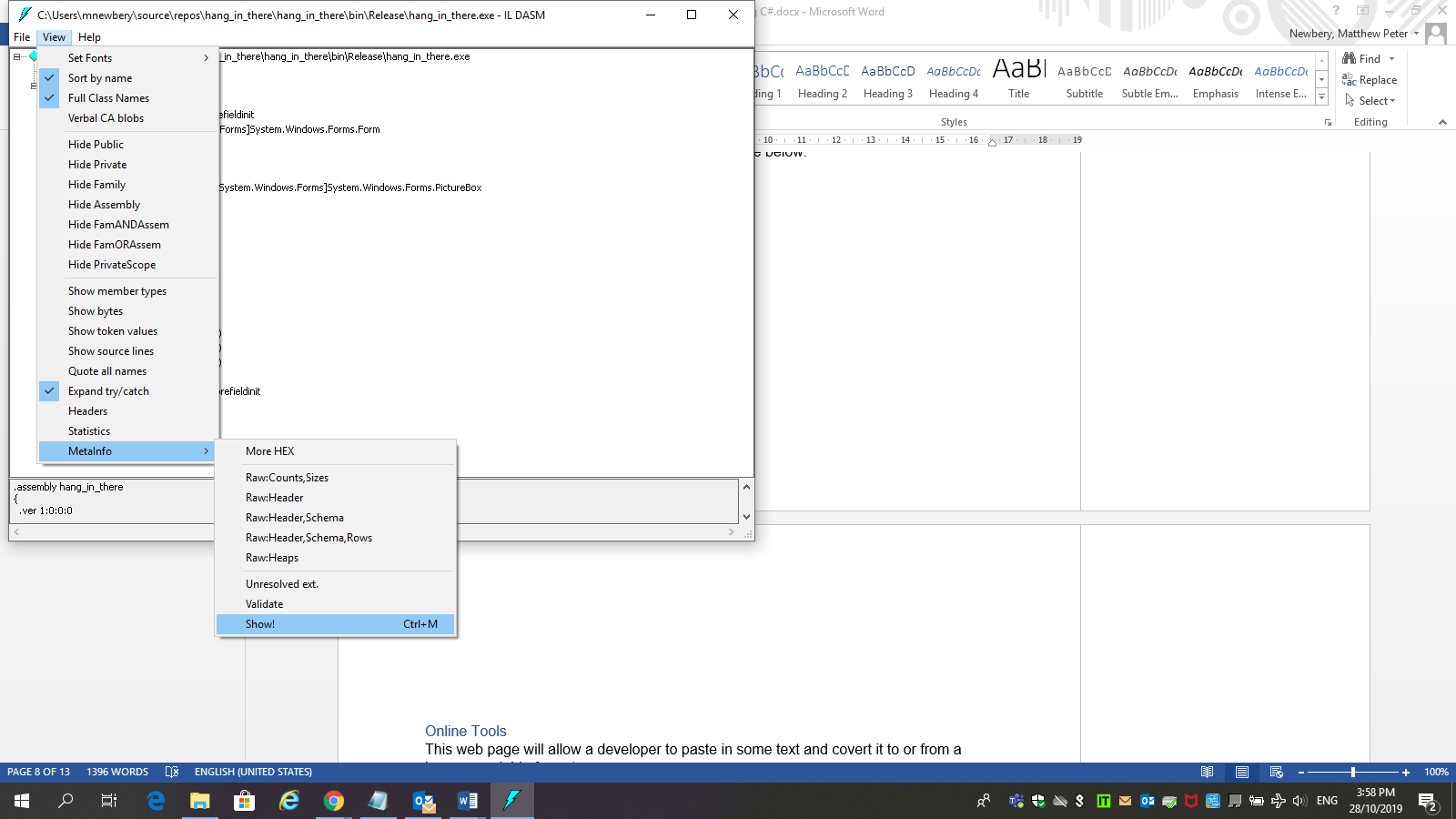
Once ILDASM is running, use this interface to open the binary file provided. See below:



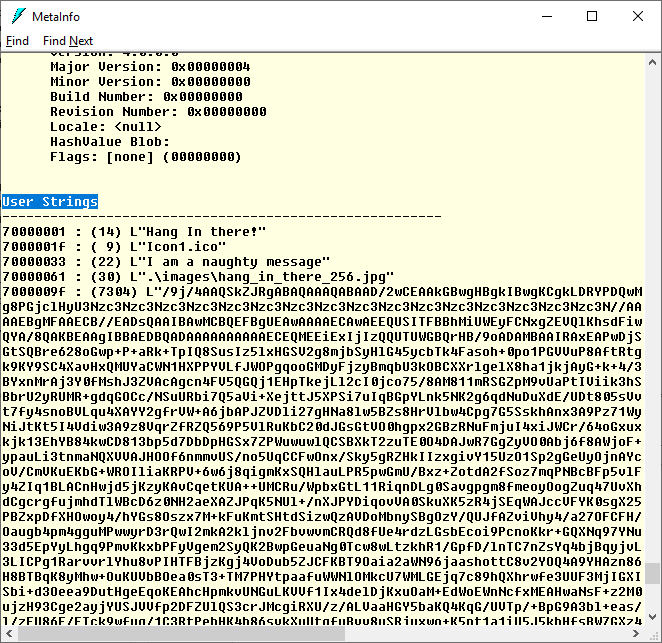
Now that the binary is open, navigate through the tree view by expanding the nodes. An example appears below:



At the top of the application window, there is a menu. Ensure the function in the previous screen grab is selected then click on the “view” menu item then select “MetaInfo > Show” from the menu that appears. See below:



View the new screen detail that appears:



Psst. Scroll Down the window!

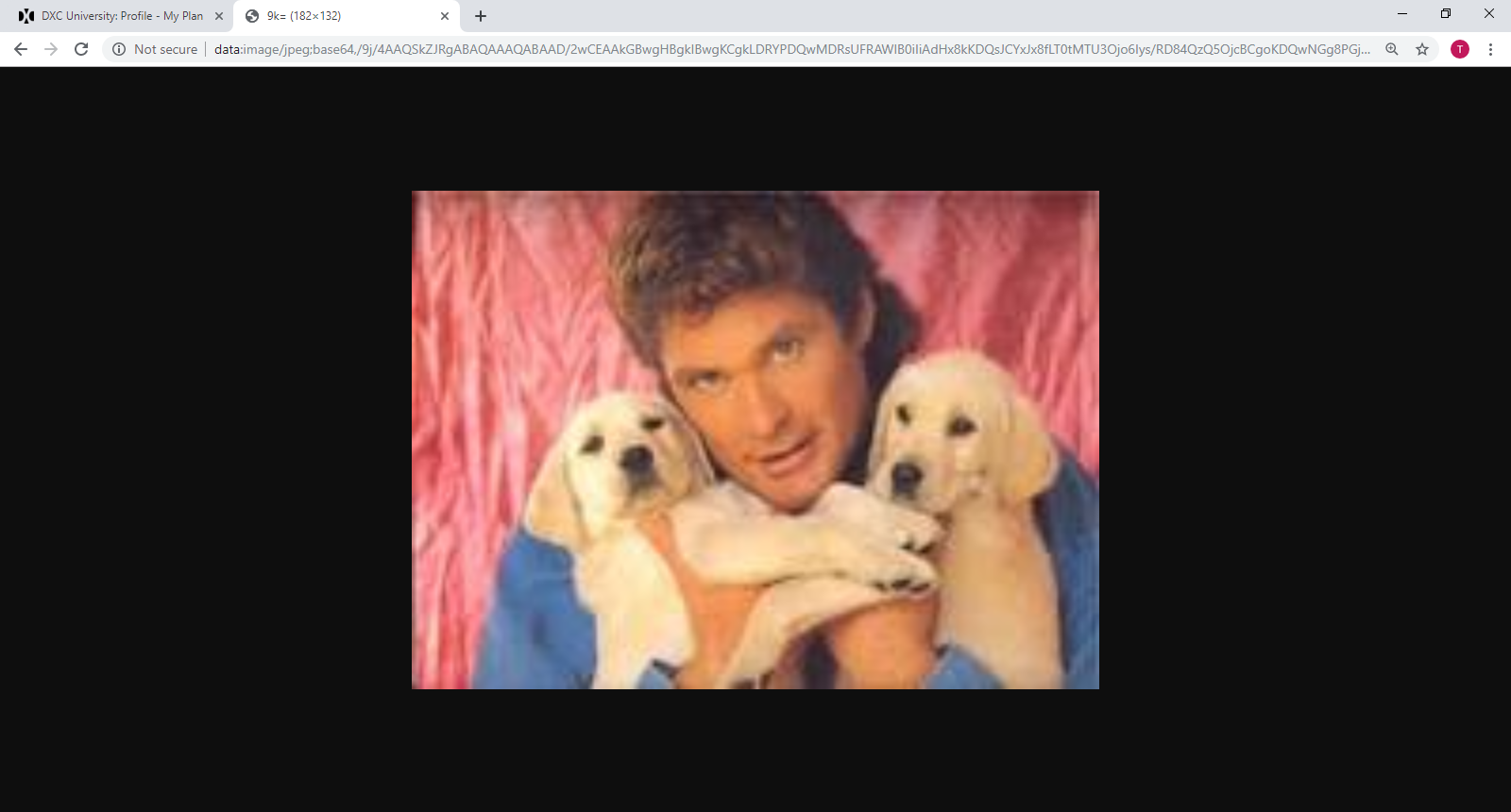
The last user string appears to be 7304 decimal characters long. It also starts with “/9j/”. The hint section of this document suggests this is the start of a base64 representation of a JPEG.

Copy the contents of this user string into a text document with notepad. Save the file.

The base64 image can be rendered in Chrome by turning the base64 string into a URL. Do this now by opening a new tab in Chrome then pasting a URL into the address bar in the following format (yes keep the comma):

data:image/jpeg;base64,<the user string from the binary>

See the example below:



Is this the image embedded in the binary?

If so, save the image from the browser as a jpeg. This image is to be included in the capture-the-flag report.

## Using Jet Brains Dot Peek

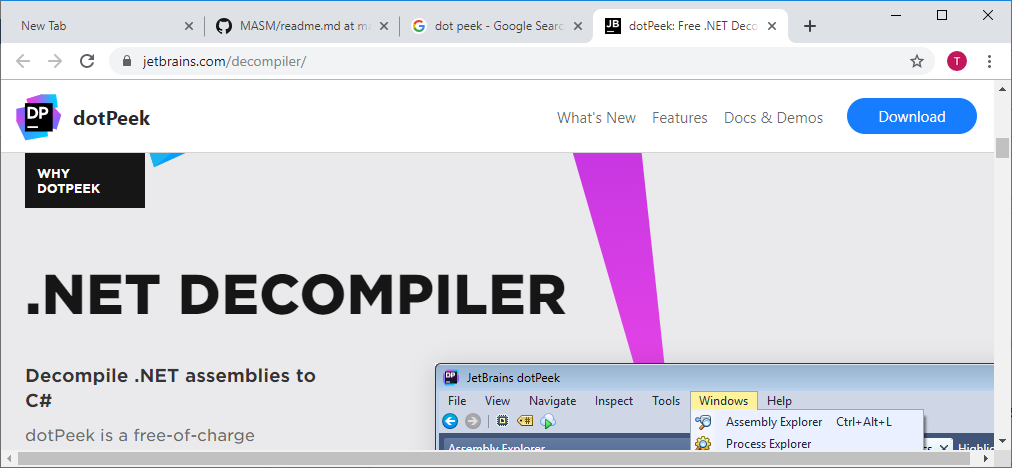
JetBrains dotPeek (Dot Peek) is a free-of-charge standalone tool based on the JetBrains ReSharper bundled decompiler. It can reliably decompile any .NET assembly into equivalent C# or IL code.

The decompiler supports multiple formats including libraries (.dll), executables (.exe), and Windows metadata files (.winmd).

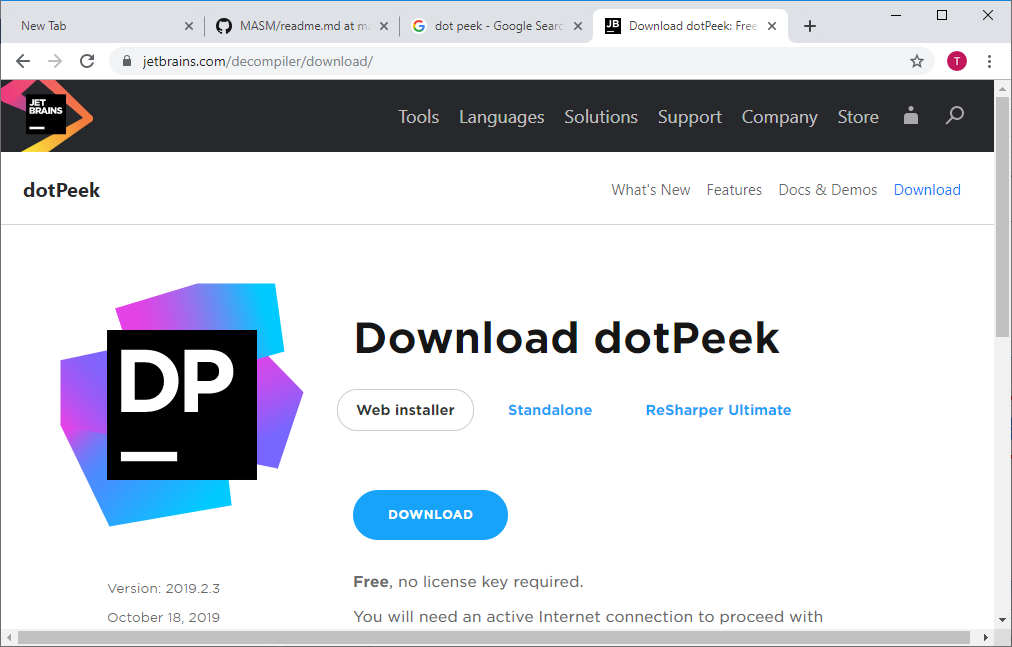
This section describes how to install and use Dot Peek to view and extract user strings from a Windows C# binary.

To get started, navigate to the JetBrains web page and download the free installer. See below:

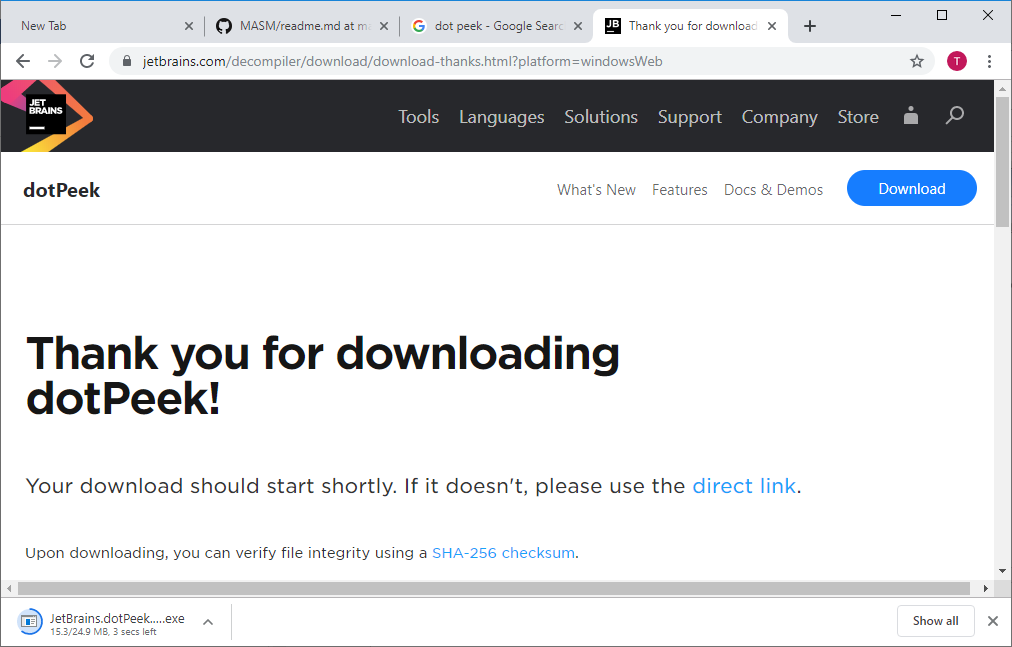
<https://www.jetbrains.com/decompiler/>



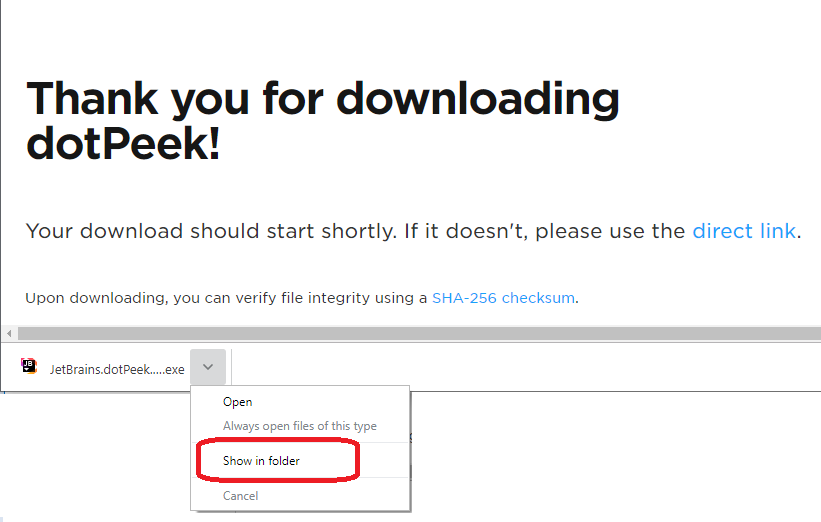
A new web page will appear. Click the download button to continue. See below:



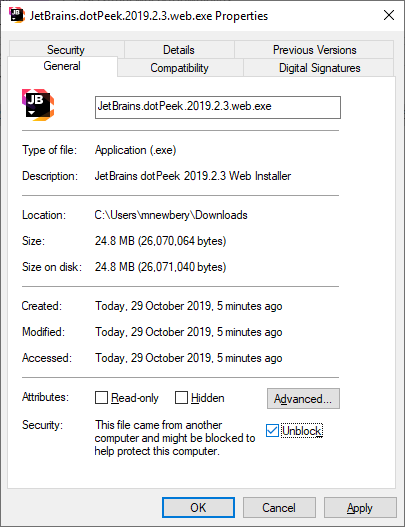
View the message noting the download should now start. The image below shows that the download is progressing. Once the download is complete, view the target folder. See below then read the next instruction:



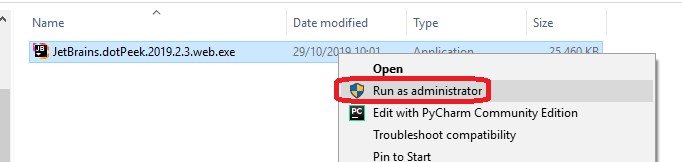
Click on the chevron to view the menu item. Select “Show in folder” from the menu that appears. See below:



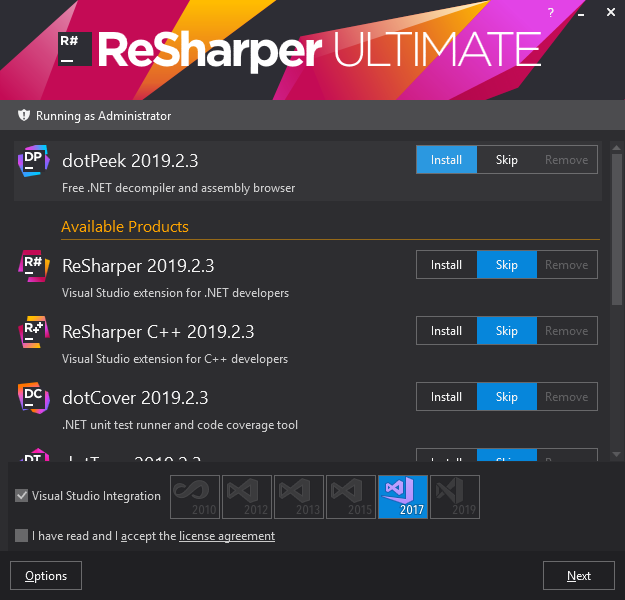
Unblock the file before continuing. See below:



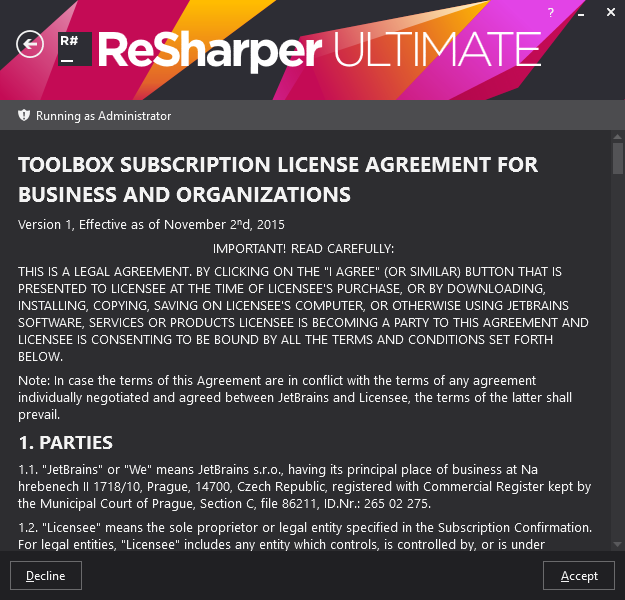
**ALWAYS RUN INSTALLERS AS ADMINISTRATOR!** Any installer may be run “as administrator” when the user right clicks on the installed then selects “Run as administrator from the menu that appears. See the image below:



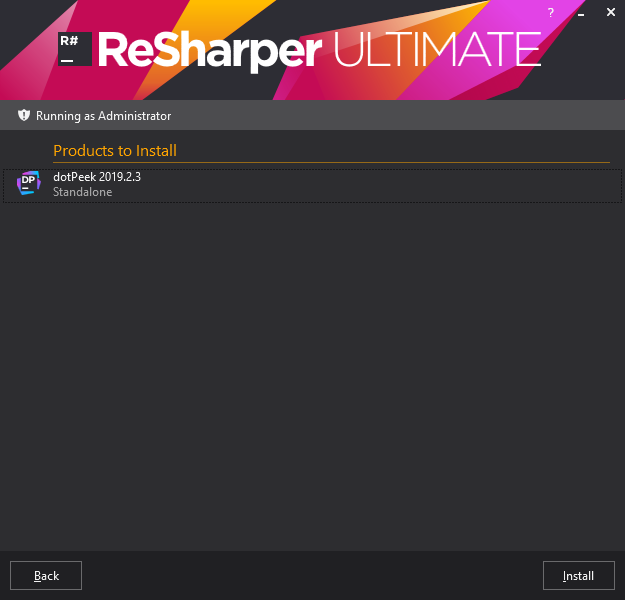
When the prompt “Do you want to allow this app to make changes to your device” appears, select “Yes” to continue. Install to the default location when prompted. Below is the next dialog to appear. Tick the box for “I have read and understood the licence agreement then click Next to continue:



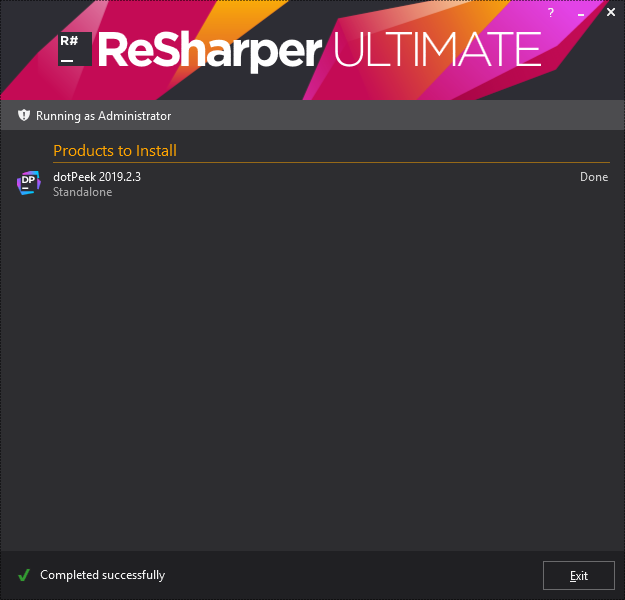
Click Accept for the licence agreement to continue. See below:



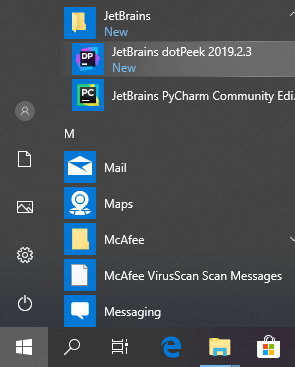
Click install to continue. See below:



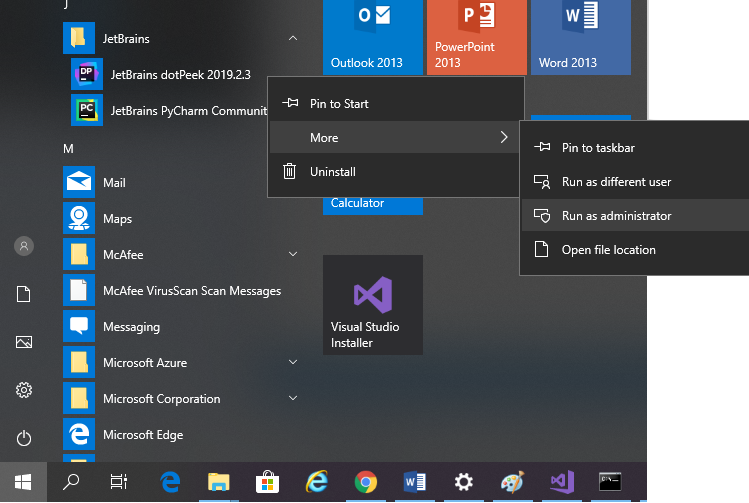
Wait for the installation to complete then click Exit to continue



A new shortcut to JetBrains will appear in the start menu. See below:



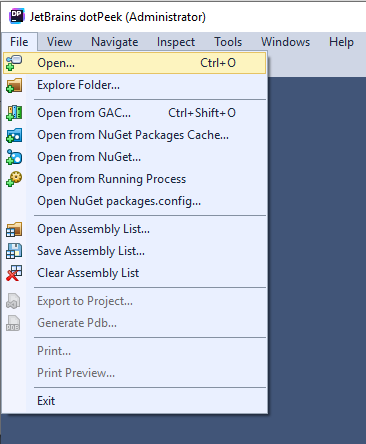
To start using Dot Peek, right click on the shortcut then choose “More > Run as administrator”. See below:



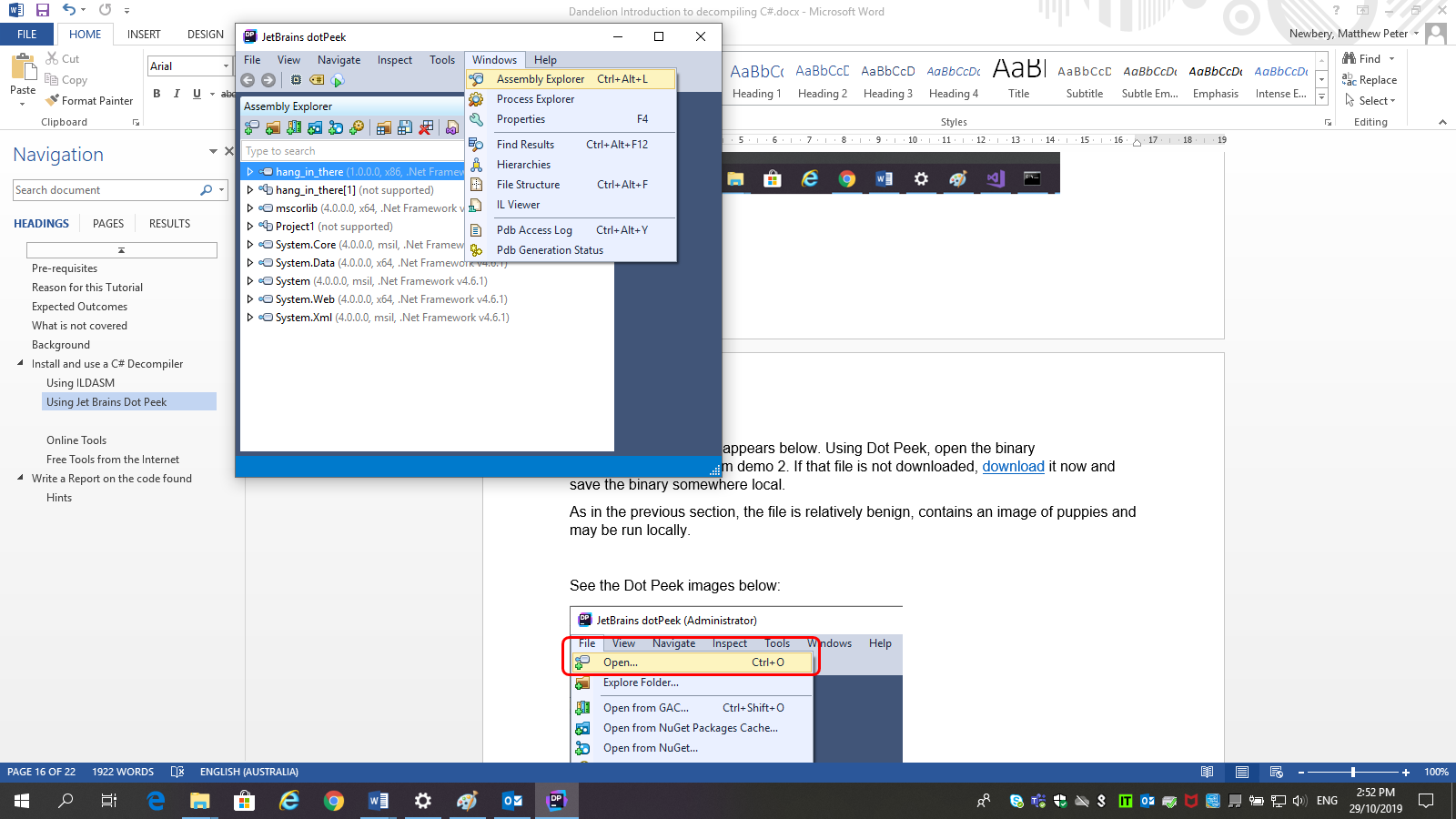
The Dot Peek interface appears below. Using Dot Peek, open the binary “hang\_in\_there.exe” from demo 2. If that file is not downloaded, [download](https://github.com/mnewbery/MASM/blob/master/demo2/hang_in_there.exe) it now and save the binary somewhere local.

As in the previous section, the file is relatively benign, contains an image of puppies and may be run locally.

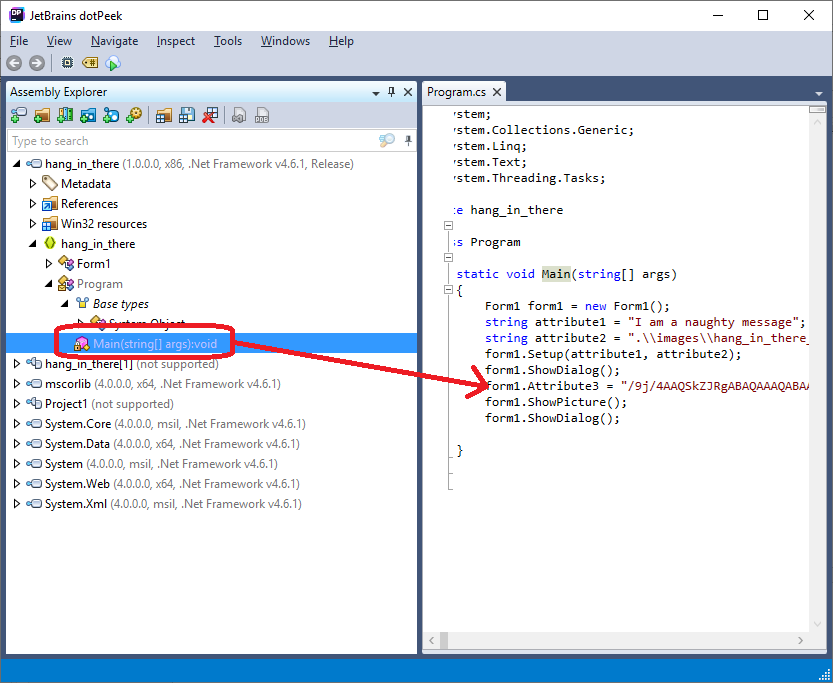
See the Dot Peek images below:



View the assembly explorer window. See below:



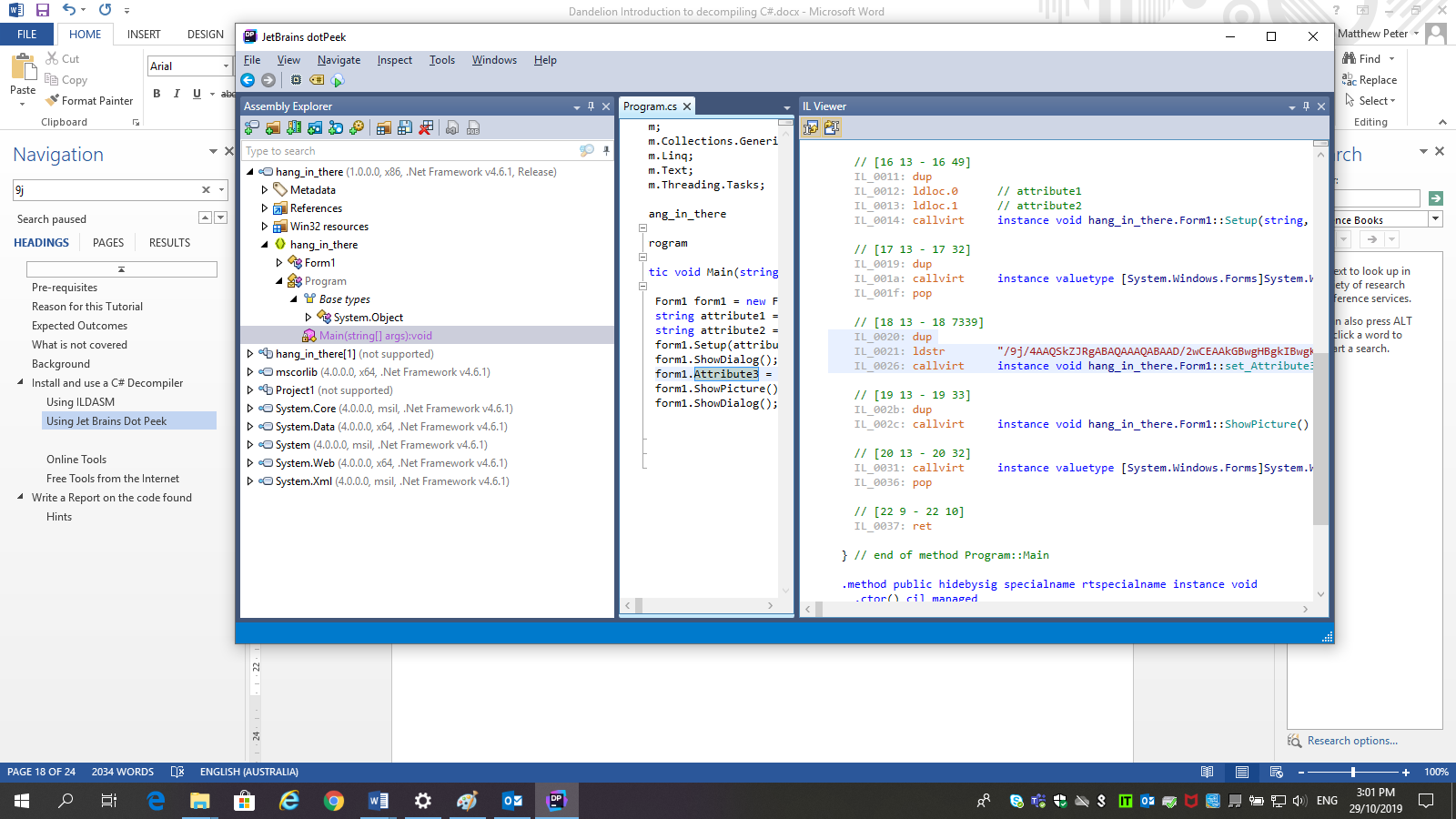
Expand the assembly tree for “hang\_in\_there.exe” then double click on the function “main.exe”. View the user strings in the code window that appears. See below:



The user string named Attribute3 appears to be 7304 decimal characters long. It also starts with “/9j/”. The hint section of this document suggests this is the start of a base64 representation of a JPEG.

Copy the contents of this user string into a text document with notepad. Save the file and check that this is the same string that was retrieved using ILDASM in the previous section.

Double click on the user string in the code window to open the IL Viewer window. See below:



In the capture-the-flag report, explain what the IL Viewer is showing for the selected user string. If in doubt, refer to the Dot Peek help files.

Follow the instructions at the end of the previous section to view the base64 image in a Chrome browser. From the browser, right click on the image then save the image as a JPEG. Attach this image to the email containing the completed capture the flag report.

Instructions for the capture-the-flag-report appear in the next section.

# Write a Report on the code found

Note: **The “Flag” that is being captured in this report** is the successful extraction of the image embedded in the binary followed by the image being sent as a JPEG attachment to the author. The report will document “The captured Flag” in the following way:

* How and when the code used the embedded image; and
* How the extraction was performed.

Complete the Capture-The-Flag report based on the sample binary file. These files can be downloaded from the GitHub repository to your local machine

<https://github.com/mnewbery/MASM/blob/master/Capture%20the%20Flag%20report%20template.docx>

https://github.com/mnewbery/MASM/blob/master/demo2/hang\_in\_there.exe

# Online Tools

This web page will allow a developer to paste in some text and covert it to or from a human readable format

<https://www.rapidtables.com/convert/number/ascii-hex-bin-dec-converter.html>

This epoch converter is handy to see what “seconds since 1970” looks like in a human readable format. The epoch converter will accept both big-endian hex and integers once punctuation is removed. Note here that time stamps in byte code are Little-Endian and may need to be converted to Big-Endian before being input to the tool

<https://www.epochconverter.com/>

## Free Tools from the Internet

Jet Brains Dot Peek free

<https://www.jetbrains.com/decompiler/download/#section=standalone>

Base64 image encoder and decoder

<https://www.base64encode.net/>

<https://www.base64decode.net/base64-image-decoder>

Step 3: Using the MASM32 tool **DumPE.exe** or by any other means, analyse the binary file. Provide answers to the following questions and note the answers in a new, blank document:

1. What address does the Portable Executable Signature start at?
2. What is the value of the Portable Executable Signature?
3. What is the target Machine value?
4. How many sections does the binary say it has?
5. What is the Time Date Stamp in hex and the human readable value?
6. Where is the Symbol Table?
7. How many symbols are in the symbol table?
8. What is the Optional header size?
9. What are the Characteristics flags that were set and what do they mean?
10. What is the Magic number?
11. What is the Linker Version?
12. What is the Target Subsystem?

Step 4: Answer these extra questions and note the answers in the same document

1. Where does the base64 image start and end in the binary?
2. Extract the base64 image, convert it to a JPEG file and send it to the tutorial author

## Hints

Jpeg file format (as many others) can be identified by magic number. For JPEG the magic number is ff d8 ff at offset 0. If you encode this to Base64, you'll always get /9j/.

Where does this string appear in the binary?

What character(s) always(s) appear at the end of a base64 value?