U.S. Army Cyber School

DEFEND ATTACK EXPLOIT

Security & Exploitation: Reverse Engineering Guides

Version Date: 26 MAR 2020

Ghidra Student Resource and Cheat Sheet

What is Ghidra?

Ghidra is a reverse engineering tool developed and released to the public by the National Security Agency. It has many capabilities to include disassembly and decompilation for many different types of executables and architectures.
Software and information related to Ghidra are hosted here: https://ghidra-sre.org/
The official Ghidra cheat sheet is located here: https://ghidra-sre.org/CheatSheet.html

Getting setup

Installation

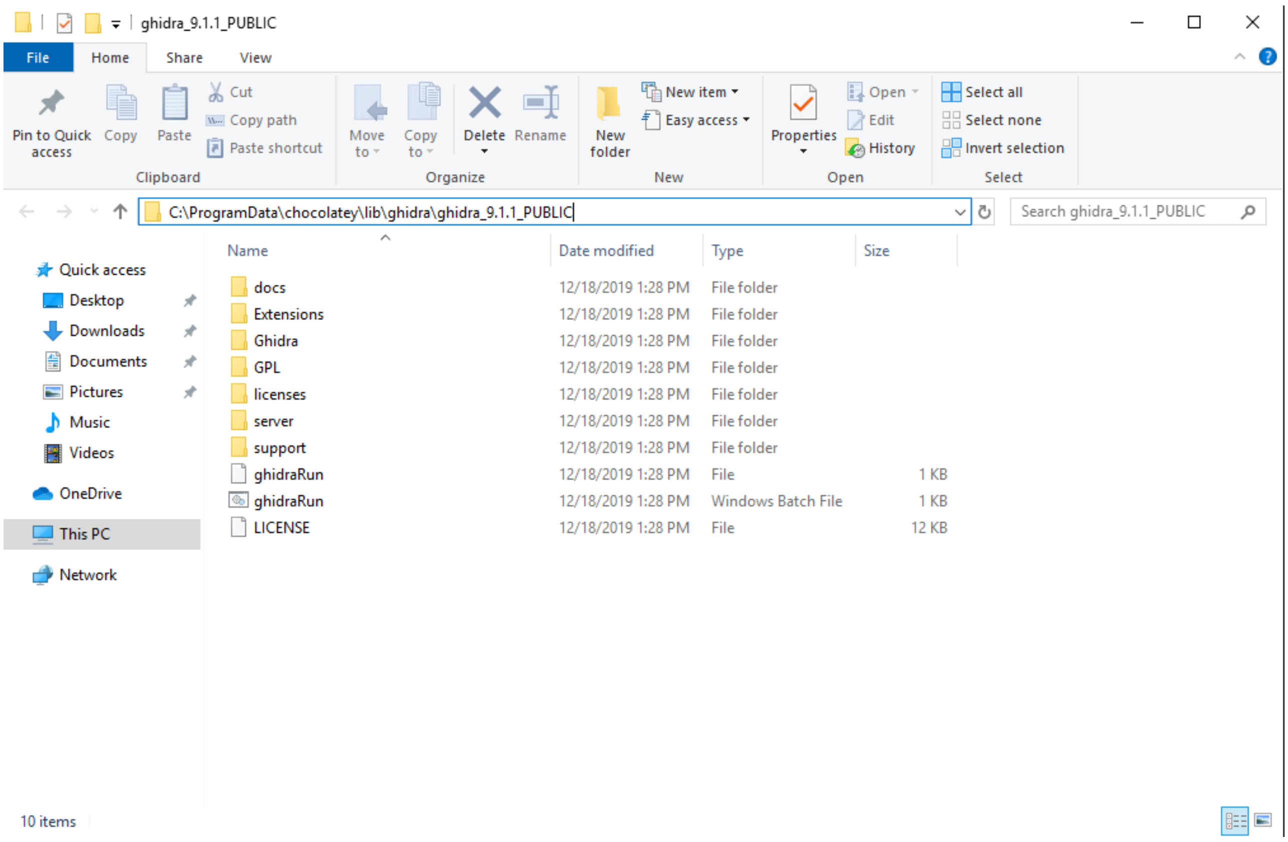
Follow the steps for your OS outlined here: https://ghidra-sre.org/InstallationGuide.html
If you are using Windows, there is a package manager called Chocolatey that can be used to automatically install Ghidra and its dependencies. Regardless of OS choice, Ghidra is built on Java. Installation will require a compatible version of Java to be installed and referenced in the PATH variable.

This guide will follow use of Ghidra on a Windows 10 machine. However, there are very few differences between running Ghidra on Linux vs Windows.

Running Ghidra

There will be a file called "ghidraRun.bat" for Windows installations. This will be located wherever you extracted Ghidra too. "ghidraRun.bat" is the startup file. It is recommended that you link this file to the Desktop or somewhere easily accessible.

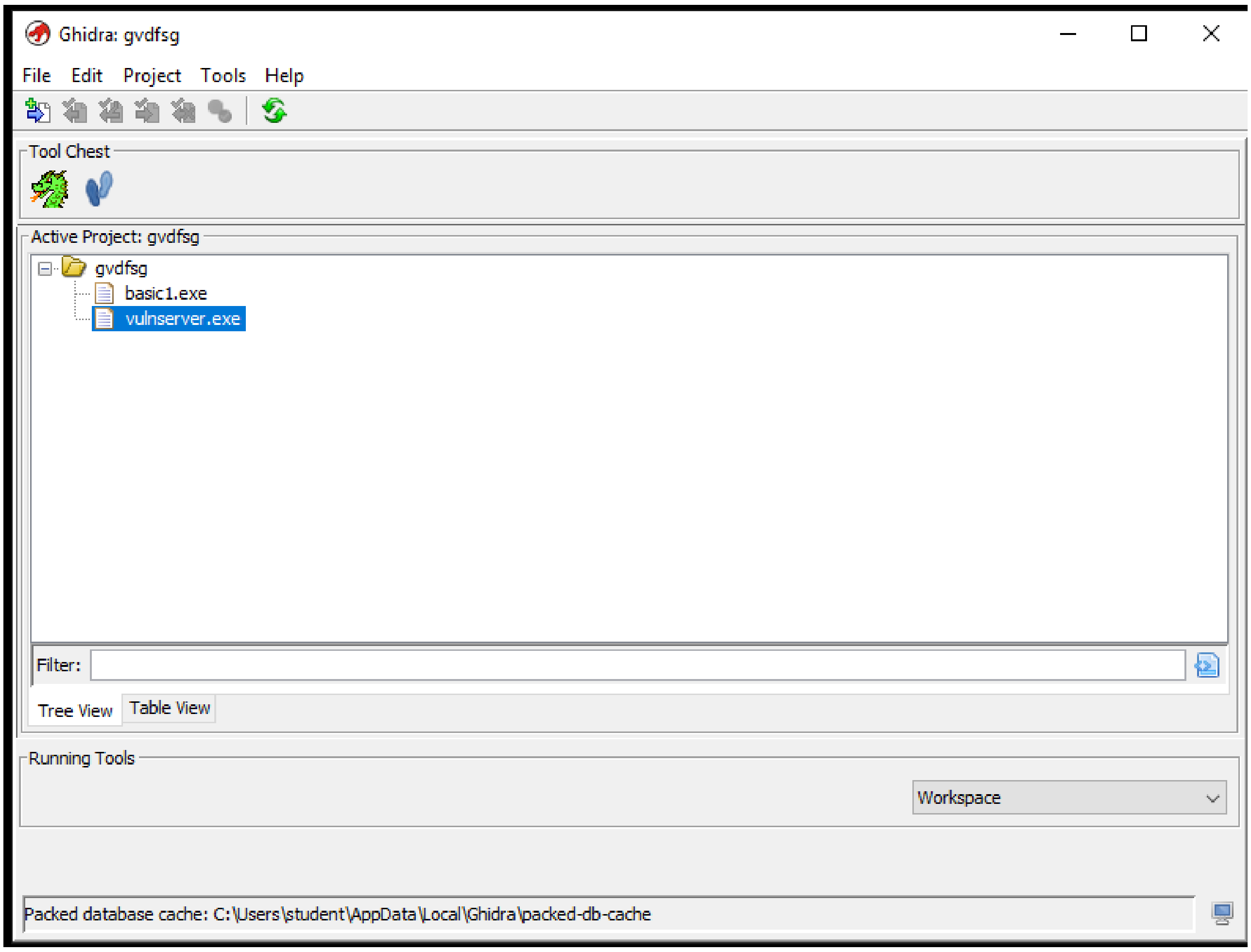
Below is a screenshot of where "ghidraRun.bat" is located when installed with Chocolatey package manager:


C:\ProgramData\chocolatey\lib\ghidra\ghidra_9.1.1_PUBLIC

Name	Date modified	Type	Size
docs	12/18/2019 1:28 PM	File folder	
Extensions	12/18/2019 1:28 PM	File folder	
Ghidra	12/18/2019 1:28 PM	File folder	
GPL	12/18/2019 1:28 PM	File folder	
licenses	12/18/2019 1:28 PM	File folder	
server	12/18/2019 1:28 PM	File folder	
support	12/18/2019 1:28 PM	File folder	
ghidraRun	12/18/2019 1:28 PM	File	1 KB
ghidraRun	12/18/2019 1:28 PM	Windows Batch File	1 KB
LICENSE	12/18/2019 1:28 PM	File	12 KB

10 items

When run, "ghidraRun.bat" will open a blank terminal for a few seconds then the Ghidra splash screen will display. After a few more seconds, the project window will open and will look like below:


Ghidra: gvdfsg

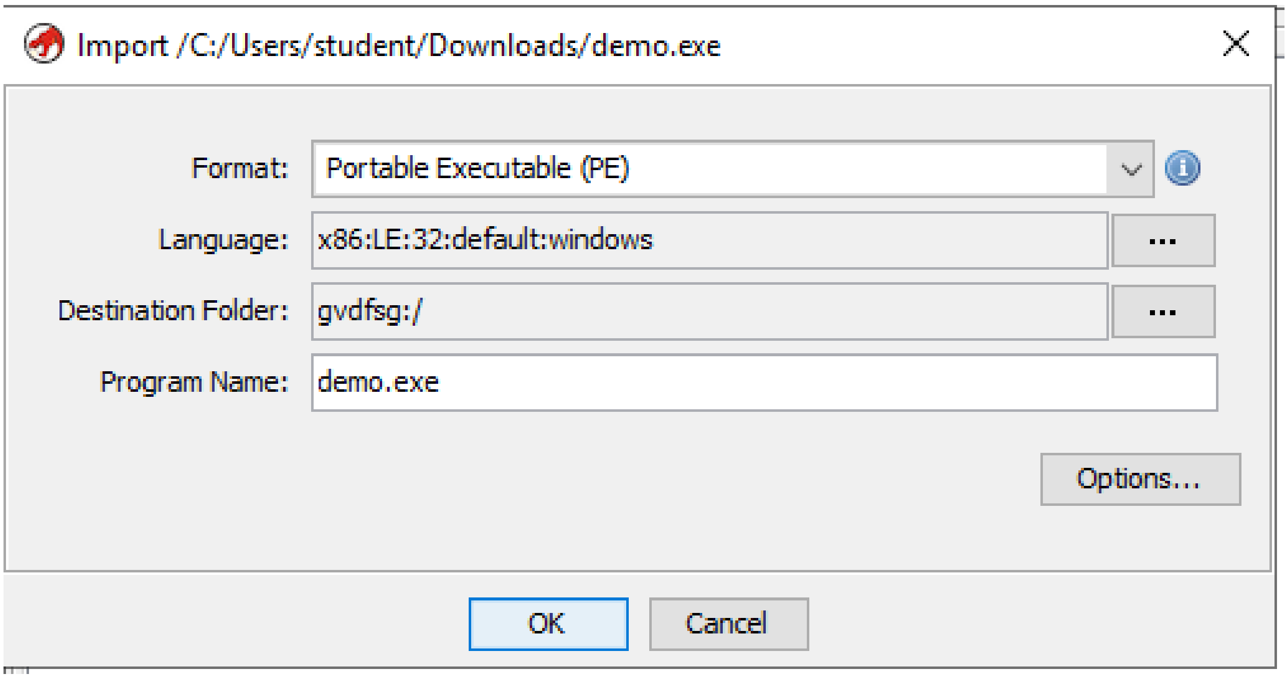
Active Project: gvdfsg

```
gvdfsg
    basic1.exe
    vulnserver.exe
```

Packed database cache: C:\Users\student\AppData\Local\Ghidra\packed-db-cache

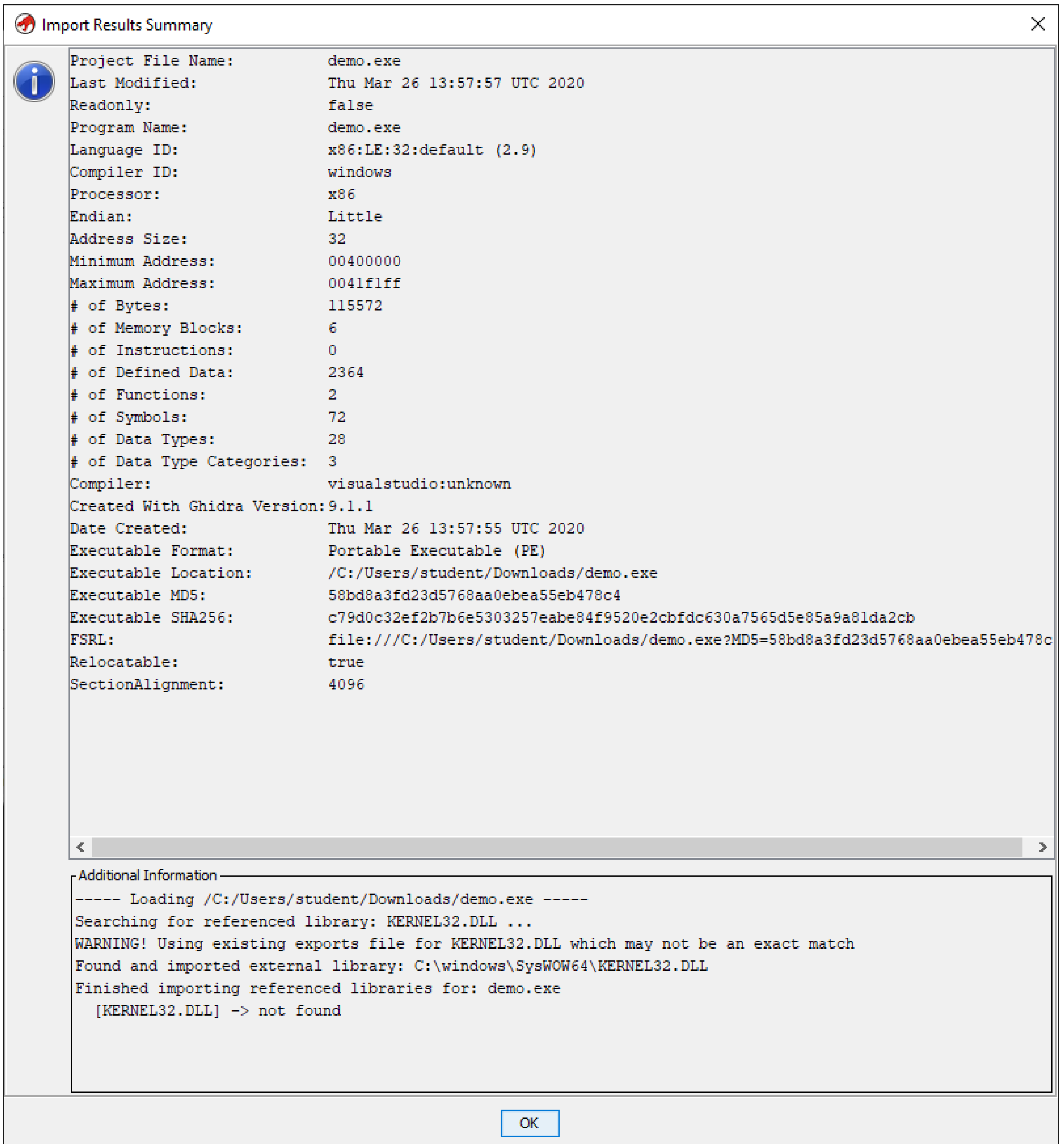
To create a project, go to File→New Project. Select "Non-shared Project", give it a name and place to be saved, fill out any additional prompts, and click finish.

Next you'll need to import a file by going to File→Import File. Select your file and it will be imported. A splash screen confirming information about the file will display. The default options will almost always be the right ones. However, you can easily verify them based off your findings during static analysis. In the below example, Ghidra properly identified the imported file as an x86, Little Endian, 32 bit, portable executable for Windows.


Import /C:/Users/student/Downloads/demo.exe

Format: Portable Executable (PE)
Language: x86:LE:32:default:windows
Destination Folder: gvdfsg:/
Program Name: demo.exe

The next splash screen will show additional information about the binary such as compiler. In the example, it correctly identified it as compiled with Visual Studio:



```
Project File Name:          demo.exe
Last Modified:              Thu Mar 26 13:57:57 UTC 2020
Readonly:                   false
Program Name:               demo.exe
Language ID:                x86:LE:32:default (2.9)
Compiler ID:                windows
Processor:                  x86
Endian:                     Little
Address Size:               32
Minimum Address:            00400000
Maximum Address:            0041f1ff
# of Bytes:                 115572
# of Memory Blocks:         6
# of Instructions:          0
# of Defined Data:          2364
# of Functions:             2
# of Symbols:               72
# of Data Types:            28
# of Data Type Categories:  3
Compiler:                   visualstudio:unknown
Created With Ghidra Version:9.1.1
Date Created:               Thu Mar 26 13:57:55 UTC 2020
Executable Format:          Portable Executable (PE)
Executable Location:        /C:/Users/student/Downloads/demo.exe
Executable MD5:             58bd8a3fd23d5768aa0ebea55eb478c4
Executable SHA256:          c79d0c32ef2b7b6e5303257eabe84f9520e2cbfdc630a7565d5e85a9a81da2cb
FSRL:                       file:///C:/Users/student/Downloads/demo.exe?MD5=58bd8a3fd23d5768aa0ebea55eb478c
Relocatable:                true
SectionAlignment:           4096
```

Additional Information

```
----- Loading /C:/Users/student/Downloads/demo.exe -----
Searching for referenced library: KERNEL32.DLL ...
WARNING! Using existing exports file for KERNEL32.DLL which may not be an exact match
Found and imported external library: C:\windows\SysWOW64\KERNEL32.DLL
Finished importing referenced libraries for: demo.exe
  [KERNEL32.DLL] -> not found
```

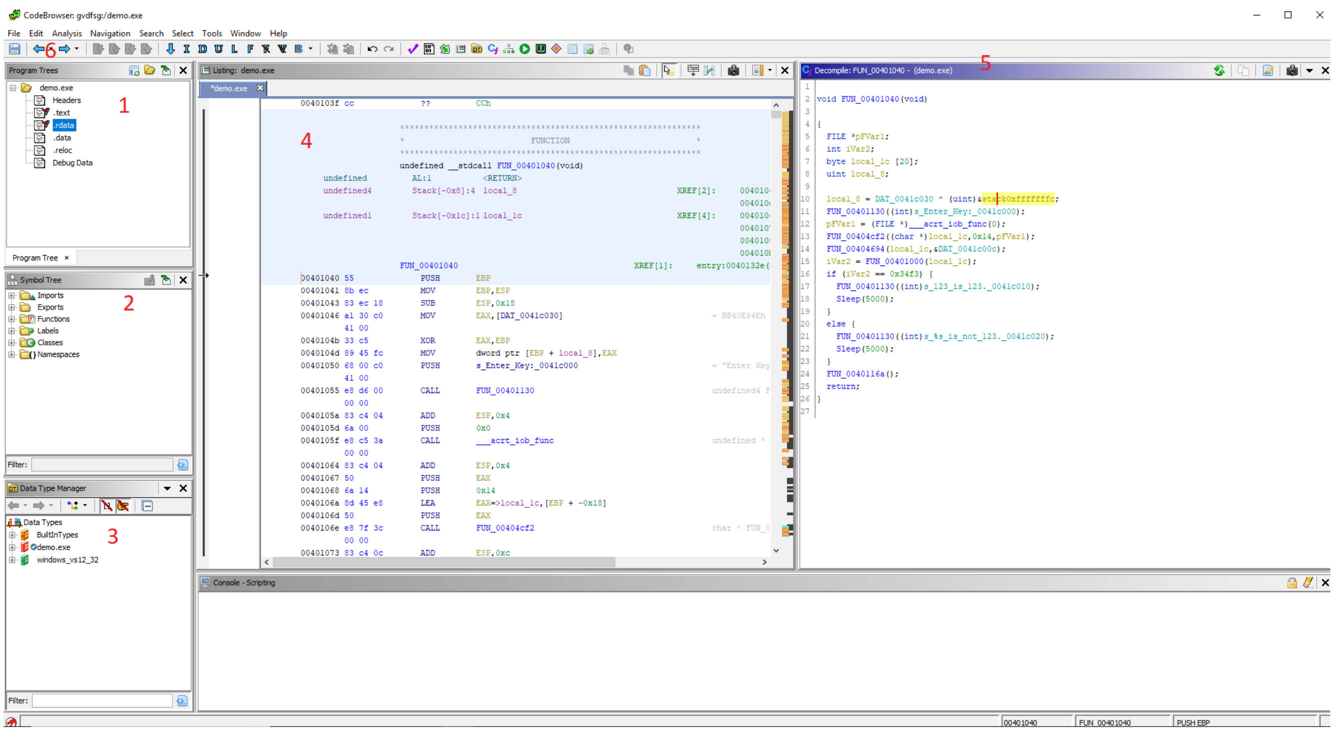
Double clicking on your imported file will automatically launch "CodeBrowser". This is the main tool used to analyze the binary. It displays the disassembly, imports, functions, decompilation, and many other things about the binary.

Begin by clicking "Yes" to analyze the binary. Otherwise, Ghidra may not be able to find most of the important things we care about as reverse engineers. Accept the defaults as they are most appropriate at an entry level. The analysis may take a few minutes. There is a status bar in the bottom right corner that will show the analysis progress.

Click "ok" if a window pops up complaining of "Incomplete PDB information". This will happen regularly, as not all binaries are compiled with deug information.

Overview of Ghira’s CodeBrowser features

The below list references the numbers on this screenshot:



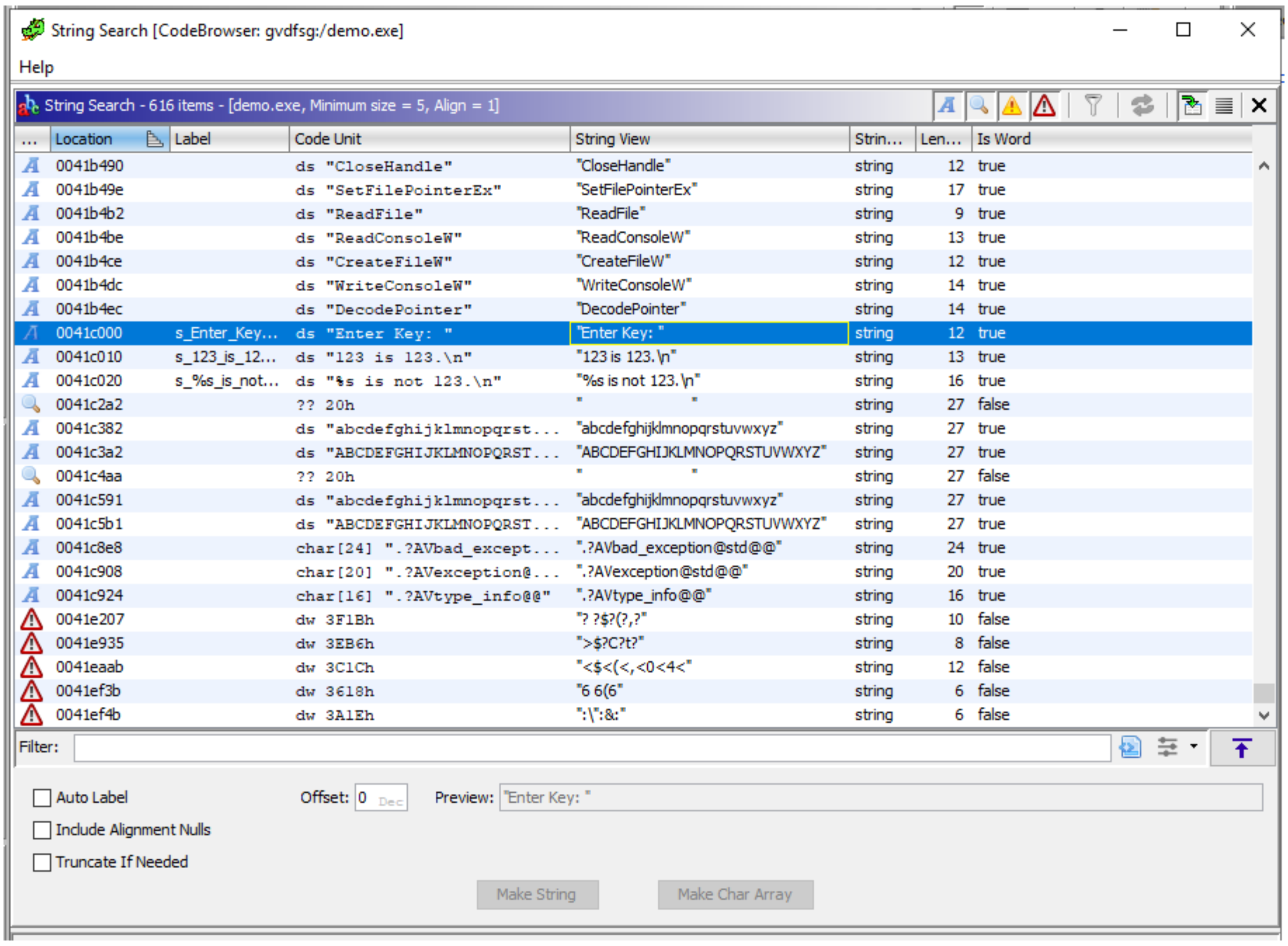
```
0040103f cc            ??          CCh

                       ****************************************************************
                       *                          FUNCTION                            *
                       ****************************************************************
                       undefined __stdcall FUN_00401040(void)
     undefined            AL:1           <RETURN>
     undefined4           Stack[-0x8]:4  local_8                        XREF[2]:    004010
                                                                                    004010
     undefined1           Stack[-0x1c]:1 local_1c                       XREF[4]:    004010
                                                                                    004010
                                                                                    004010
                                                                                    004010
                       FUN_00401040                              XREF[1]:    entry:0040132e(
00401040 55            PUSH        EBP
00401041 8b ec         MOV         EBP,ESP
00401043 83 ec 18      SUB         ESP,0x18
00401046 a1 30 c0      MOV         EAX,[DAT_0041c030]                        = BB40E64Eh
         41 00
0040104b 33 c5         XOR         EAX,EBP
0040104d 89 45 fc      MOV         dword ptr [EBP + local_8],EAX
00401050 68 00 c0      PUSH        s_Enter_Key:_0041c000                     = "Enter Key
         41 00
00401055 e8 d6 00      CALL        FUN_00401130                              undefined4 F
         00 00
0040105a 83 c4 04      ADD         ESP,0x4
0040105d 6a 00         PUSH        0x0
0040105f e8 c5 3a      CALL        ___acrt_iob_func                          undefined *
         00 00
00401064 83 c4 04      ADD         ESP,0x4
00401067 50            PUSH        EAX
00401068 6a 14         PUSH        0x14
0040106a 8d 45 e8      LEA         EAX=>local_1c,[EBP + -0x18]
0040106d 50            PUSH        EAX
0040106e e8 7f 3c      CALL        FUN_00404cf2                              char * FUN_0
         00 00
00401073 83 c4 0c      ADD         ESP,0xc
```

```

void FUN_00401040(void)

{
  FILE *pFVar1;
  int iVar2;
  byte local_1c [20];
  uint local_8;

  local_8 = DAT_0041c030 ^ (uint)&stack0xfffffffc;
  FUN_00401130((int)s_Enter_Key:_0041c000);
  pFVar1 = (FILE *)___acrt_iob_func(0);
  FUN_00404cf2((char *)local_1c,0x14,pFVar1);
  FUN_00404694(local_1c,&DAT_0041c00c);
  iVar2 = FUN_00401000(local_1c);
  if (iVar2 == 0x34f3) {
    FUN_00401130((int)s_123_is_123._0041c010);
    Sleep(5000);
  }
  else {
    FUN_00401130((int)s_%s_is_not_123._0041c020);
    Sleep(5000);
  }
  FUN_0040116a();
  return;
}

```

1. Section headers - This allows you to jump to specific sections of the program in disassembly
2. Symbol Tree - This allows you to dig through the Imports, Exports, Functions, etc. It can be very handy to dig into functions as sometimes the names are retained and can be very meaningful.
3. Data Type Manager - Used to organize data types used in the binary and during analysis.
4. Disassembly - Shows the disassembled binary. If it is an x86 binary, the assembly code will be displayed in a syntax very close to Intel.
5. Decompiler - Displays pseudo-C++ decompiled code. It is generally accurate, but sometimes it will miss things that can be easily seen in the disassembly. Do not trust this blindly. Always confirm or dig deeper with the disassembly and other analysis techniques.
6. Back/Forward Button - These buttons take you backward, or forward, one step from where you are. So, if you entered a function call and are done analyzing it, you can press back and it will take you back to where the function call happened. You may have to press back more than once depending on how much you did when inside the function call.

Finding strings

To find strings within the disassembled binary go to the top menu and select Search→For Strings. The defaults are generally ok, but many analysts will remove the "Require Null Termination" and drop the minimum length to 3. This is not necessary at an entry level. Select search when satisfied with settings. This will pop up a box like the one below:


String Search [CodeBrowser: gvdfsg:/demo.exe]
Help
String Search - 616 items - [demo.exe, Minimum size = 5, Align = 1]

...	Location	Label	Code Unit	String View	Strin...	Len...	Is Word
A	0041b490		ds "CloseHandle"	"CloseHandle"	string	12	true
A	0041b49e		ds "SetFilePointerEx"	"SetFilePointerEx"	string	17	true
A	0041b4b2		ds "ReadFile"	"ReadFile"	string	9	true
A	0041b4be		ds "ReadConsoleW"	"ReadConsoleW"	string	13	true
A	0041b4ce		ds "CreateFileW"	"CreateFileW"	string	12	true
A	0041b4dc		ds "WriteConsoleW"	"WriteConsoleW"	string	14	true
A	0041b4ec		ds "DecodePointer"	"DecodePointer"	string	14	true
A	0041c000	s_Enter_Key...	ds "Enter Key: "	"Enter Key: "	string	12	true
A	0041c010	s_123_is_12...	ds "123 is 123.\n"	"123 is 123.\n"	string	13	true
A	0041c020	s_%s_is_not...	ds "%s is not 123.\n"	"%s is not 123.\n"	string	16	true
🔍	0041c2a2		?? 20h	" "	string	27	false
A	0041c382		ds "abcdefghijklmnopqrst...	"abcdefghijklmnopqrstuvwxyz"	string	27	true
A	0041c3a2		ds "ABCDEFGHIJKLMNOPQRST...	"ABCDEFGHIJKLMNOPQRSTUVWXYZ"	string	27	true
🔍	0041c4aa		?? 20h	" "	string	27	false
A	0041c591		ds "abcdefghijklmnopqrst...	"abcdefghijklmnopqrstuvwxyz"	string	27	true
A	0041c5b1		ds "ABCDEFGHIJKLMNOPQRST...	"ABCDEFGHIJKLMNOPQRSTUVWXYZ"	string	27	true
A	0041c8e8		char[24] ".?AVbad_except...	".?AVbad_exception@std@@"	string	24	true
A	0041c908		char[20] ".?AVexception@...	".?AVexception@std@@"	string	20	true
A	0041c924		char[16] ".?AVtype_info@@"	".?AVtype_info@@"	string	16	true
⚠	0041e207		dw 3F1Bh	"? ?$?(?,?"	string	10	false
⚠	0041e935		dw 3EB6h	">$?C?t?"	string	8	false
⚠	0041eaab		dw 3C1Ch	"<$<(<,<0<4<"	string	12	false
⚠	0041ef3b		dw 3618h	"6 6(6"	string	6	false
⚠	0041ef4b		dw 3A1Eh	":\":&:"	string	6	false

Filter:

- [ ] Auto Label
- [ ] Include Alignment Nulls
- [ ] Truncate If Needed

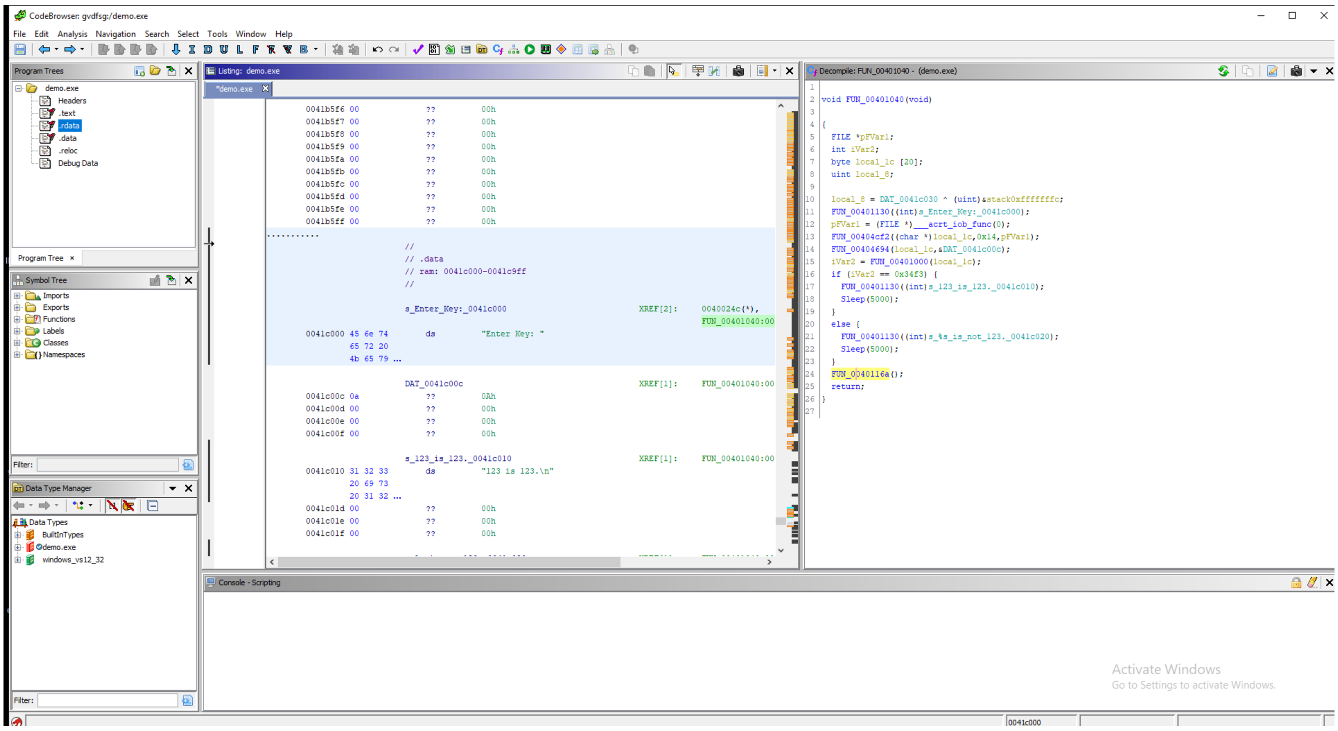
Offset: 0 Dec

Preview: "Enter Key: "

These are all the strings that were found in the binary based on your settings. The search bar at the bottom of this window is called "Filter:". It will find strings containing what you type into it.

In the above picture, a string "Enter Key" is highlighted by the analyst. Double clicking on this will take them to the location in the data section where this resides. Once double clicked, the analyst will have to select the original CodeBrowser window again, as the Strings window will not automatically hide itself and will remain covering the disassembly/data window.

The screenshot below shows where the "Enter Key: " string is located in data:

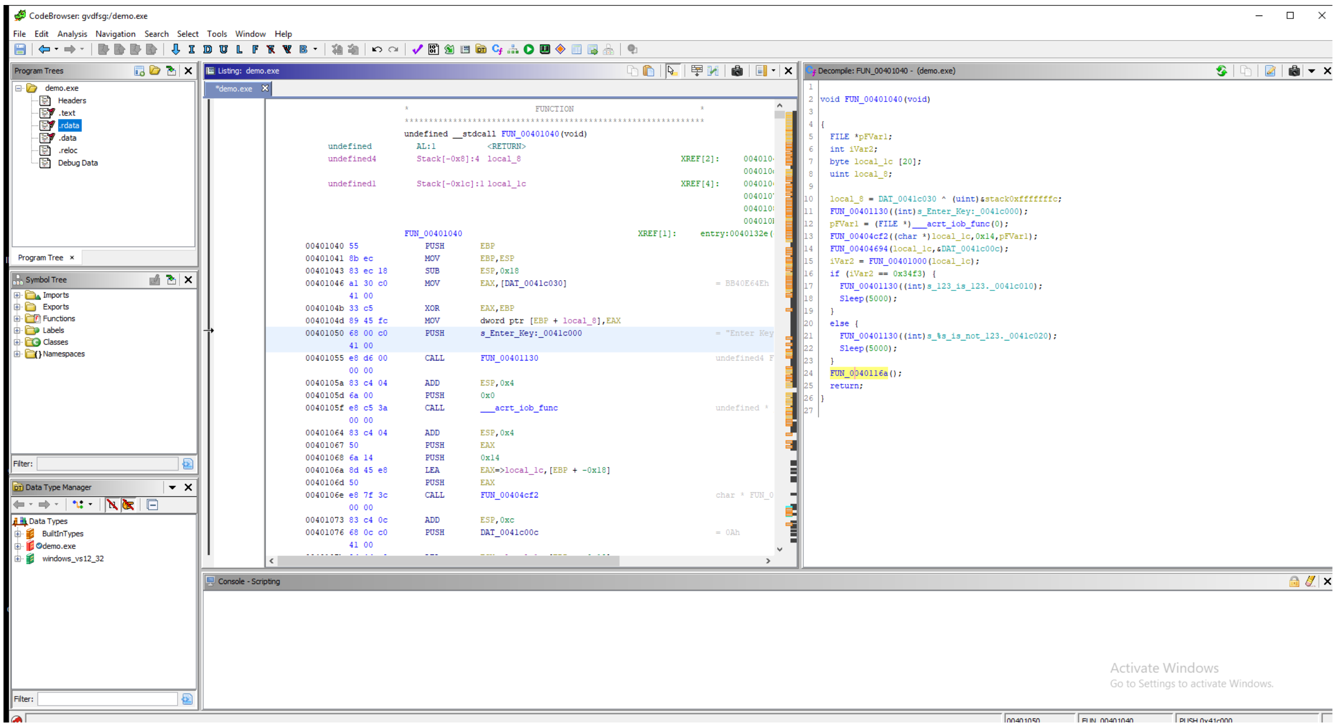


```
0041b5f6 00          ??         00h
0041b5f7 00          ??         00h
0041b5f8 00          ??         00h
0041b5f9 00          ??         00h
0041b5fa 00          ??         00h
0041b5fb 00          ??         00h
0041b5fc 00          ??         00h
0041b5fd 00          ??         00h
0041b5fe 00          ??         00h
0041b5ff 00          ??         00h
...........
                     //
                     // .data
                     // ram: 0041c000-0041c9ff
                     //

                     s_Enter_Key:_0041c000                    XREF[2]:     0040024c(*),
                                                                            FUN_00401040:00
0041c000 45 6e 74       ds         "Enter Key: "
         65 72 20
         4b 65 79 ...

                     DAT_0041c00c                             XREF[1]:     FUN_00401040:00
0041c00c 0a          ??         0Ah
0041c00d 00          ??         00h
0041c00e 00          ??         00h
0041c00f 00          ??         00h

                     s_123_is_123._0041c010                   XREF[1]:     FUN_00401040:00
0041c010 31 32 33       ds         "123 is 123.\n"
         20 69 73
         20 31 32 ...
0041c01d 00          ??         00h
0041c01e 00          ??         00h
0041c01f 00          ??         00h
```

```c

void FUN_00401040(void)

{
  FILE *pFVar1;
  int iVar2;
  byte local_1c [20];
  uint local_8;

  local_8 = DAT_0041c030 ^ (uint)&stack0xfffffffc;
  FUN_00401130((int)s_Enter_Key:_0041c000);
  pFVar1 = (FILE *)___acrt_iob_func(0);
  FUN_00404cf2((char *)local_1c,0x14,pFVar1);
  FUN_00404694(local_1c,&DAT_0041c00c);
  iVar2 = FUN_00401000(local_1c);
  if (iVar2 == 0x34f3) {
    FUN_00401130((int)s_123_is_123._0041c010);
    Sleep(5000);
  }
  else {
    FUN_00401130((int)s_%s_is_not_123._0041c020);
    Sleep(5000);
  }
  FUN_0040116a();
  return;
}

```

The analyst has already highlighted the function that is referencing the "Enter Key: " string. Because of this selection, the decompiler has brought that function up into view.

Double clicking on the highlighted function reference will take us to that disassembly which looks like the screenshot below:



```
                     *                        FUNCTION                        *
                     **********************************************************
                     undefined __stdcall FUN_00401040(void)
      undefined         AL:1           <RETURN>
      undefined4        Stack[-0x8]:4  local_8                        XREF[2]:     004010
                                                                                  004010
      undefined1        Stack[-0x1c]:1 local_1c                       XREF[4]:     004010
                                                                                  004010
                                                                                  004010
                                                                                  004010

                     FUN_00401040                             XREF[1]:     entry:0040132e(
00401040 55             PUSH       EBP
00401041 8b ec          MOV        EBP,ESP
00401043 83 ec 18       SUB        ESP,0x18
00401046 a1 30 c0       MOV        EAX,[DAT_0041c030]                        = BB40E64Eh
         41 00
0040104b 33 c5          XOR        EAX,EBP
0040104d 89 45 fc       MOV        dword ptr [EBP + local_8],EAX
00401050 68 00 c0       PUSH       s_Enter_Key:_0041c000                     = "Enter Key
         41 00
00401055 e8 d6 00       CALL       FUN_00401130                              undefined4 F
         00 00
0040105a 83 c4 04       ADD        ESP,0x4
0040105d 6a 00          PUSH       0x0
0040105f e8 c5 3a       CALL       ___acrt_iob_func                          undefined *
         00 00
00401064 83 c4 04       ADD        ESP,0x4
00401067 50             PUSH       EAX
00401068 6a 14          PUSH       0x14
0040106a 8d 45 e8       LEA        EAX=>local_1c,[EBP + -0x18]
0040106d 50             PUSH       EAX
0040106e e8 7f 3c       CALL       FUN_00404cf2                              char * FUN_0
         00 00
00401073 83 c4 0c       ADD        ESP,0xc
00401076 68 0c c0       PUSH       DAT_0041c00c                              = 0Ah
         41 00
```

```c

void FUN_00401040(void)

{
  FILE *pFVar1;
  int iVar2;
  byte local_1c [20];
  uint local_8;

  local_8 = DAT_0041c030 ^ (uint)&stack0xfffffffc;
  FUN_00401130((int)s_Enter_Key:_0041c000);
  pFVar1 = (FILE *)___acrt_iob_func(0);
  FUN_00404cf2((char *)local_1c,0x14,pFVar1);
  FUN_00404694(local_1c,&DAT_0041c00c);
  iVar2 = FUN_00401000(local_1c);
  if (iVar2 == 0x34f3) {
    FUN_00401130((int)s_123_is_123._0041c010);
    Sleep(5000);
  }
  else {
    FUN_00401130((int)s_%s_is_not_123._0041c020);
    Sleep(5000);
  }
  FUN_0040116a();
  return;
}

```

Modifying variables and numbers

Ghidra's CodeBrowser allows for changing the name of disassembled objects. To change a variable or function's name, right click on it and select "Rename Function" or "Rename Variable" as appropriate. You can also select the function or variable and press "L". This is the shortcut for renaming an object.

To convert a number from hex to decimal, you must select it in the disassembly view. Right click it, and select Convert→Unsigned Decimal.

There are several other conversion options depending on the situation. These include the option to convert to character which can be very useful if single characters are being passed at a time.

Additional Capabilities

This is a very condensed explanation of some of the main functionality of Ghidra. It can do many other things based on an analyst’s needs. To learn more, go through the documentation located within your Ghidra installation directory which will look something like "ghidra\ghidra_9.1.1_PUBLIC\docs".