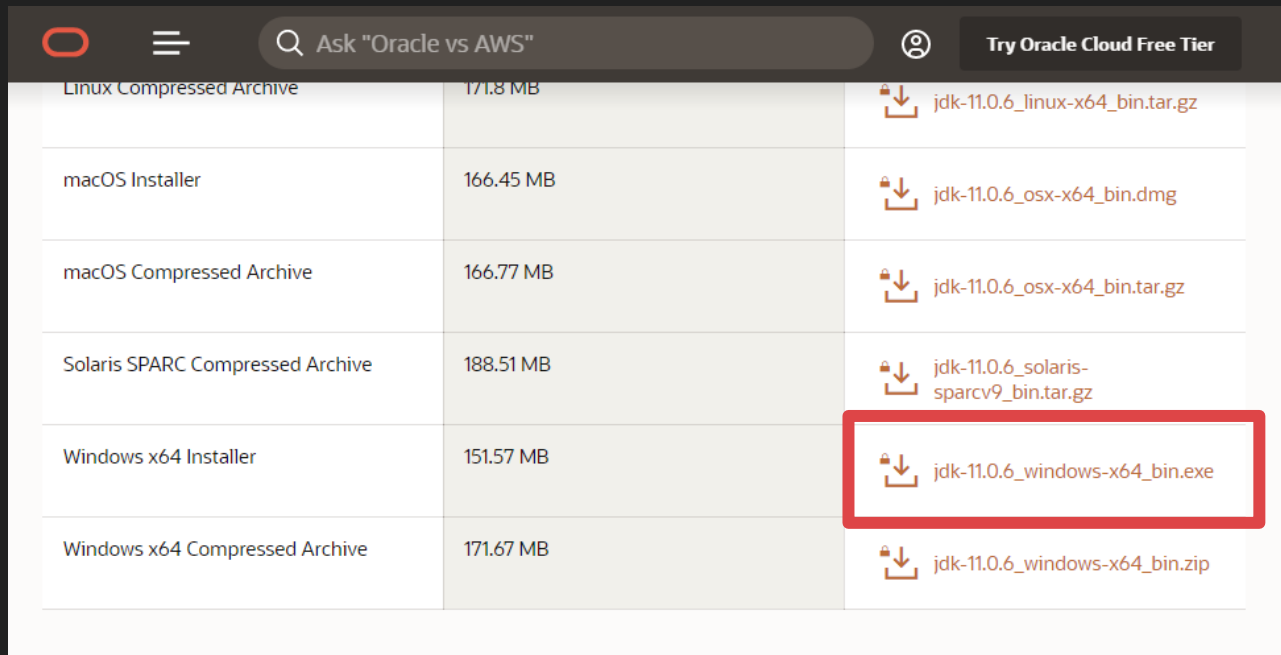








# BinDiff Tutorial

0856069 Weichun, Lin

# Installation: Java

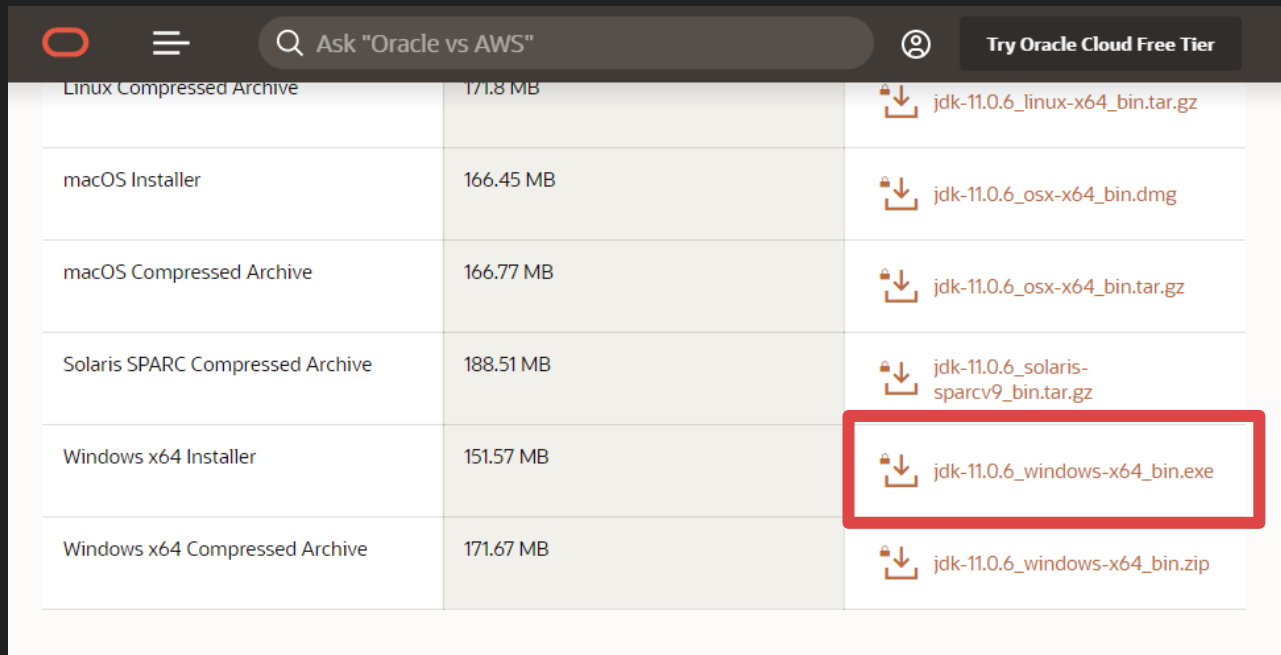
- Download Java environment from following link (An Oracle account required) :  
<https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>









Linux Compressed Archive	171.8 MB	 <a href="#">jdk-11.0.6_linux-x64_bin.tar.gz</a>
macOS Installer	166.45 MB	 <a href="#">jdk-11.0.6_osx-x64_bin.dmg</a>
macOS Compressed Archive	166.77 MB	 <a href="#">jdk-11.0.6_osx-x64_bin.tar.gz</a>
Solaris SPARC Compressed Archive	188.51 MB	 <a href="#">jdk-11.0.6_solaris-sparcv9_bin.tar.gz</a>
Windows x64 Installer	151.57 MB	 <a href="#">jdk-11.0.6_windows-x64_bin.exe</a>
Windows x64 Compressed Archive	171.67 MB	 <a href="#">jdk-11.0.6_windows-x64_bin.zip</a>

# Installation: Java

- Download Java environment from following link (An Oracle account required) :  
<https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>



Linux Compressed Archive	171.8 MB	 <a href="#">jdk-11.0.6_linux-x64_bin.tar.gz</a>
macOS Installer	166.45 MB	 <a href="#">jdk-11.0.6_osx-x64_bin.dmg</a>
macOS Compressed Archive	166.77 MB	 <a href="#">jdk-11.0.6_osx-x64_bin.tar.gz</a>
Solaris SPARC Compressed Archive	188.51 MB	 <a href="#">jdk-11.0.6_solaris-sparcv9_bin.tar.gz</a>
Windows x64 Installer	151.57 MB	 <a href="#">jdk-11.0.6_windows-x64_bin.exe</a>
Windows x64 Compressed Archive	171.67 MB	 <a href="#">jdk-11.0.6_windows-x64_bin.zip</a>

# Download and extract Ghidra

- Download Ghidra from following link: <https://ghidra-sre.org/>



# Download BinExport from GitHub

- `wget https://github.com/google/binexport/archive/master.zip`
- `wget https://github.com/google/binexport/releases/download/v11/binexport11-windows.zip`
- `unzip master.zip`
- `unzip binexport11-windows.zip`
- Ensure Java version is 11 via command `java -version` OR set PATH env variable:
  - `set PATH=C:\Program Files\Java\jdk-11.0.6\bin;%PATH%`
  - `set JAVA_HOME=C:\Program Files\Java\jdk-11.0.6`
- Fix symbol link:
  - `del binexport-master\java\BinExport\src\main\proto\binexport2.proto`
  - `mklink /H binexport-master\java\BinExport\src\main\proto\binexport2.proto Y:\binexport\binexport-master\binexport2.proto` (absolute path required)

# Download Gradle

- `wget https://downloads.gradle-dn.com/distributions/gradle-5.6.3-all.zip`
- `unzip gradle-5.6.3-all.zip`
- Add gradle to PATH via: `set PATH=Y:\binexport\gradle-5.6.3\bin;%PATH%`
- Test gradle is work correctly

```
Y:\binexport>gradle --version

-----
Gradle 5.6.3
-----

Build time:   2019-10-18 00:28:36 UTC
Revision:     bd168bbf5d152c479186a897f2cea494b7875d13

Kotlin:       1.3.41
Groovy:       2.5.4
Ant:          Apache Ant(TM) version 1.9.14 compiled on March 12 2019
JVM:          11.0.6 (Oracle Corporation 11.0.6+8-LTS)
OS:           Windows 10 10.0 amd64

Y:\binexport>
```

# Compile BinExport

- Locate where Ghidra installed via set system environment GHIDRA\_INSTALL\_DIR:
  - set GHIDRA\_INSTALL\_DIR=Y:\ghidra\_9.1.2\_PUBLIC
- cd binexport-master\java\BinExport
- gradle
- Build file located at folder: dist

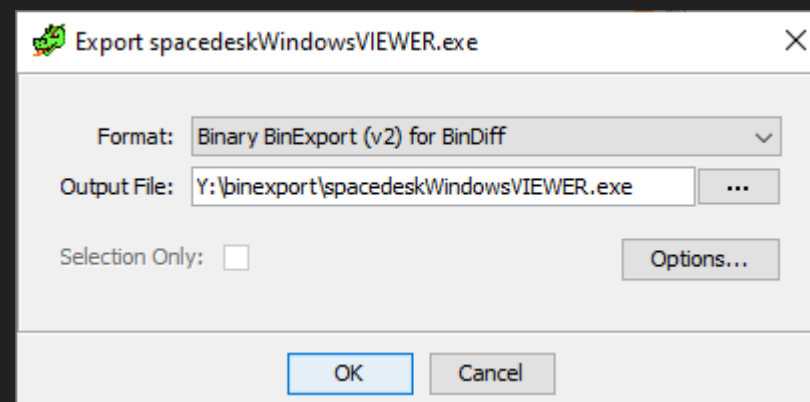
# Install Extension on Ghidra

- [File]/[Install Extensions]
- Click [plus] button and select built zip file.
- Restart Ghidra

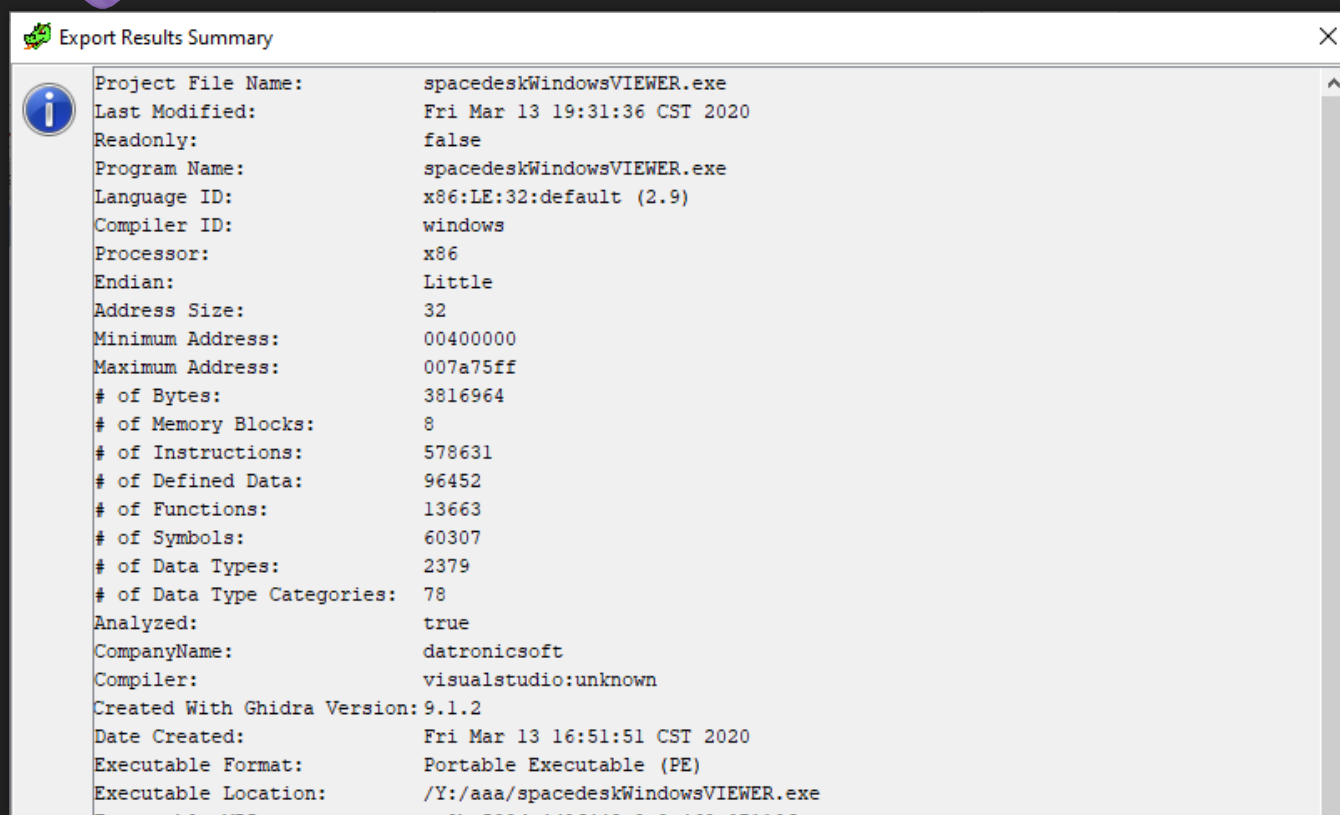


# Export file on Ghidra

- After we restart Ghidra and reopen project, we can export project as BinExport format
- [File]/[Export Program]
- Select Binary BinExport (v2) for BinDiff

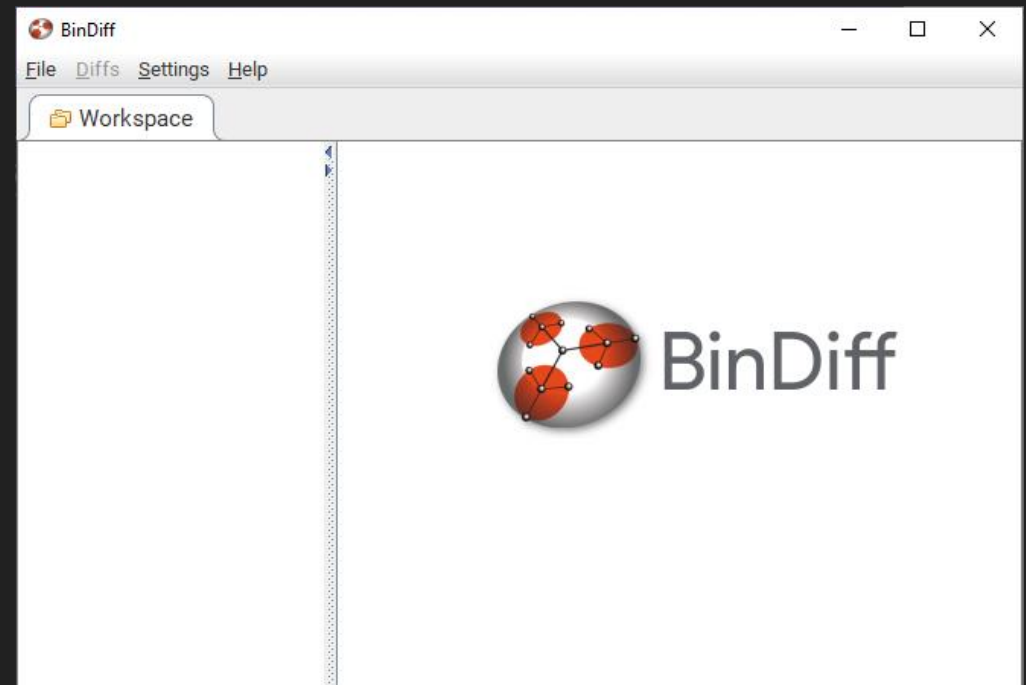


# Export file on Ghidra



# Install BinDiff

- Download BinDiff **v5** from <https://storage.googleapis.com/bindiff-releases/bindiff5.msi>
- Install BinDiff
- Open BinDiff via `bindiff_ui.cmd`



# Compare two program

```
#include <stdio.h>
int test(int a);
int main(){
    int x;
    scanf("%d", &x);
    if(test(x)){
        printf("odd\n");
    }else{
        printf("even\n");
    }
    return 0;
}
```

```
a.c
int test(int a) {
    int c = a & 1;
    return c;
}
```

```
b.c
int test(int a) {
    int c = a % 1;
    return c;
}
```

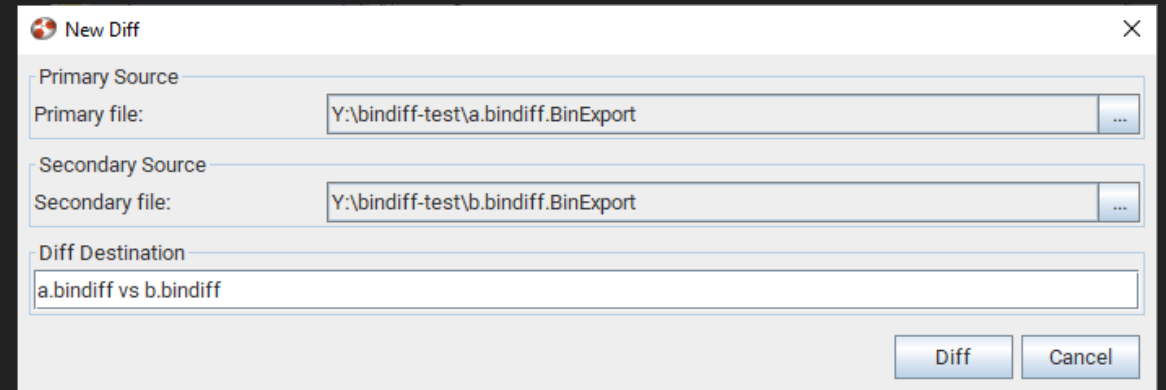
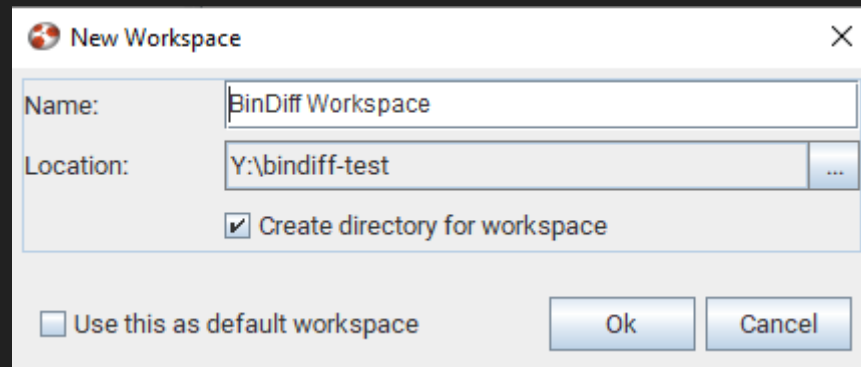
# Compile two source dump its.

- `clang a.c -o a.exe & clang b.c -o b.exe`
- Dump its via BinDiff

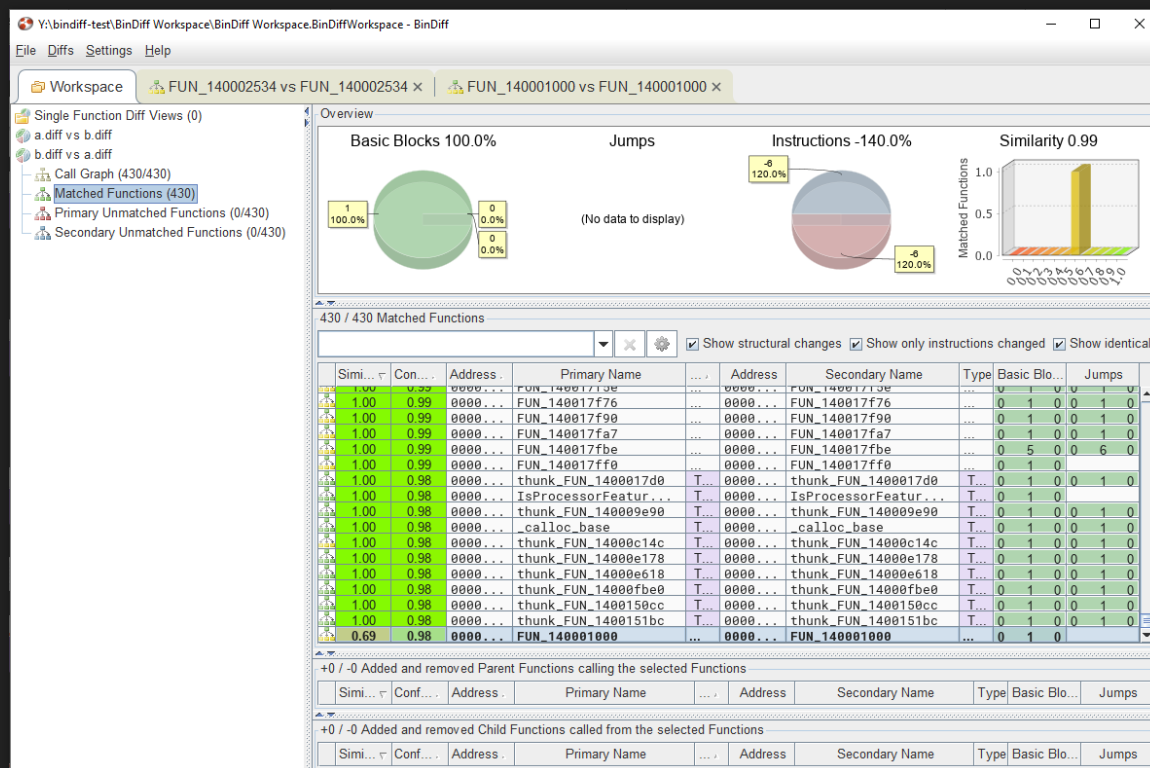
Name	Date modified	Type	Size
BinDiff Workspace	2020/03/13 19:47	File folder	
a.c	2020/03/13 19:41	C Source	1 KB
a.exe	2020/03/13 19:43	Application	144 KB
b.c	2020/03/13 19:41	C Source	1 KB
b.exe	2020/03/13 19:43	Application	144 KB
b.bindiff.BinExport	2020/03/13 19:46	BINEXPORT File	672 KB
a.bindiff.BinExport	2020/03/13 19:46	BINEXPORT File	672 KB

# Create workspace in BinDiff

- [File]/[New Workspace]
- [Diff]/[New Diff]



# See Report in BinDiff



# See Report in BinDiff

- Click on Label: [Similarity] sort it by similarity.
- Double click on last row and see this function in ASM.

The screenshot shows the BinDiff application interface with two panes: 'primary' (left) and 'secondary' (right). Both panes display the same assembly code for function FUN\_140001000, which is highlighted in yellow. The code is as follows:

```
0000000140001000 FUN_140001000
0000000140001000 PUSH RAX
0000000140001001 MOV dword ptr [RSP + local_4], ECX
0000000140001005 MOV EAX, dword ptr [RSP + local_4]
0000000140001009 CDQ
000000014000100A MOV ECX, 0x1
000000014000100F IDIV ECX
0000000140001011 MOV dword ptr [RSP], EDX
0000000140001014 MOV ECX, dword ptr [RSP]
0000000140001017 MOV EAX, ECX
0000000140001019 POP RCX
000000014000101A RET
```

The secondary pane also shows the same code, but with a different set of instructions highlighted in blue:

```
0000000140001000 FUN_140001000
0000000140001000 PUSH RAX
0000000140001001 MOV dword ptr [RSP + local_4], ECX
0000000140001005 MOV EAX, dword ptr [RSP + local_4]
0000000140001009 AND EAX, 0x1
000000014000100C MOV dword ptr [RSP], EAX
000000014000100F MOV EAX, dword ptr [RSP]
0000000140001012 POP RCX
0000000140001013 RET
```



# Check if program used some library

- Prerequisites:
  - Program should use static link
- Let's we checking a program: navicat-keygen.exe is using OpenSSL library
- We compare this program with two dll:
  - libcrypto-1\_1-x64.dll (From OpenSSL)
  - avutil-56.dll (From ffmpeg)

# Check if program used some library

- In the result we can see:
  - There are 2321 matched function in libcrypto-1\_1-x64.dll and navicat-keygen.exe
  - There are 723 matched function in avutil-56.dll and navicat-keygen.exe
- We can guess navicat-keygen.exe used OpenSSL because there are many functions in libcrypto-1\_1-x64.dll provided in OpenSSL.

