深信服技术认证之F5隐写工具初探

原创

sangfor_edu 于 2022-04-02 14:02:18 发布 3939 收藏
文章标签： 网络安全

什么是“隐写”

“隐写”，即人们通常所说的信息隐藏，是利用人类感觉器官的不敏感性（感觉冗余），以及多媒体数字信号本身存在的冗余（数据冗余特性），将秘密信息隐藏于载体信号之中的技术。秘密信息被隐藏后，不易被察觉并且不影响载体信号的感官效果和使用价值。秘密信息可以是文本、声音或图像等，载体信号可以是数字图像、文本、音频和视频文件等。隐写技术的不易察觉性，保证了隐藏信息在传输过程中不会引起注意，从而降低了秘密通信被发现的可能。

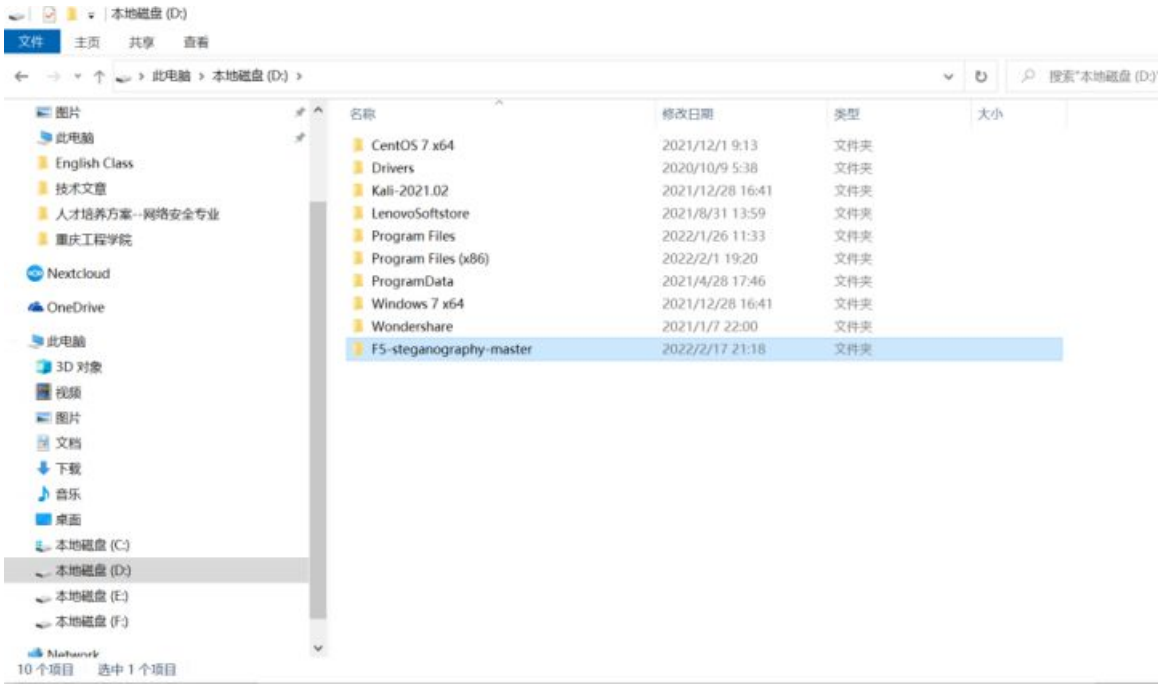
隐写的首要目标是隐蔽，也就是使加入秘密信息后的公开信息的降质尽可能小，使人无法觉察出隐藏的数据。隐写提供了一种有别于加密的安全模式，其目的不同于传统加密，不在于限制正常的数据使用，而在于保证隐藏的数据不被侵犯和觉察。隐写技术与传统的加密技术的区别关键在于：传统的加密技术仅仅隐藏了信息的内容，而隐写技术不但隐藏了信息的内容而且隐藏了信息的存在。

举一个简单的例子：在BMP图片中，一个8bit的数据代表一个像素点，在这样的图片中，人眼是无法识别11111111和11111110所代表的红色的。这里做一个定义：用来隐藏秘密信息（或信息块）的载体的二进制最小位数，称为“最低有效位”（The Least Significant Bit，LSB）。上述BMP图片的例子中，LSB即为8。

F5隐写工具的使用

在这里，我尝试着在Windows 10系统上运行该工具。

一、在GitHub上下载F5隐写工具，下载链接为https://github.com/matthewgao/F5-steganography。下载完成后解压（本例中，我将其解压至D盘根目录）：



编辑切换为居中

添加图片注释，不超过 140 字（可选）

打开D:\F5-steganography-master文件夹，其中几个主要的程序（文件）为：

（1）Embed.java：用于将信息嵌入图像文件的主程序；

（2）Extract.java：用于将隐藏信息从图像文件载体中提取的主程序；

（3）bin.noise：该文件含有需要隐藏的信息；

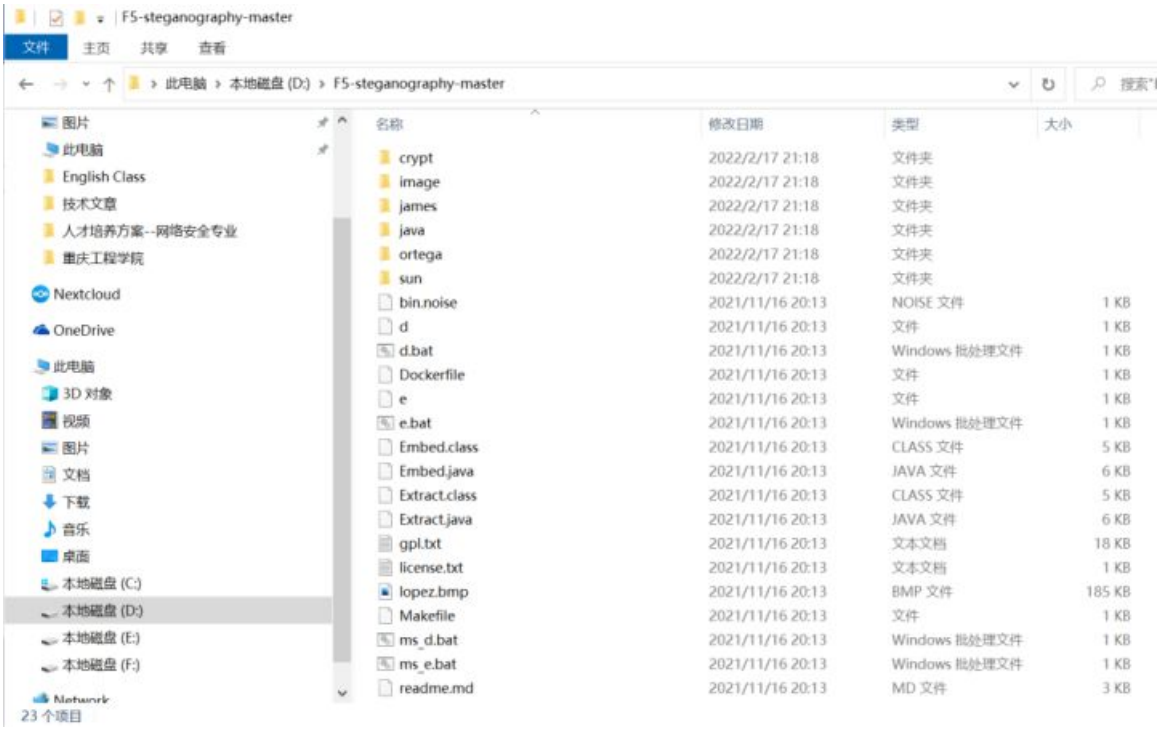
（4）e：将指定信息（bin.noise）嵌入指定图像文件（lopez.bmp）的演示程序脚本；

（5）e.bat：将指定信息（bin.noise）嵌入指定图像文件（lopez.bmp）的批处理演示程序；

（6）d：将隐藏信息从图像文件（lopez.jpg）中提取到指定文件（output.txt）的演示程序脚本；

（7）d.bat：将隐藏信息从图像文件（lopez.jpg）中提取到指定文件（output.txt）的批处理演示程序。

以上程序均可用记事本程序（notepad.exe）打开查看内容。



编辑切换为居中

添加图片注释，不超过 140 字（可选）

二、安装Java运行环境

由于F5隐写工具的两个主程序Embed.java、Extract.java均是Java语言编写的，为了正常运行它们，需要在Windows中安装Java程序运行环境。可以选择安装JDK（Java Development Kit，Java程序开发包）或JRE（Java Runtime Environment，Java程序运行环境）。本例中选择安装JDK 1.8。

（1）安装JDK 1.8

下载JDK 1.8安装包（jdk-8u191-windows-x64.exe），并按照提示安装即可。本例中，JDK安装的默认路径为C:\Program Files\Java。

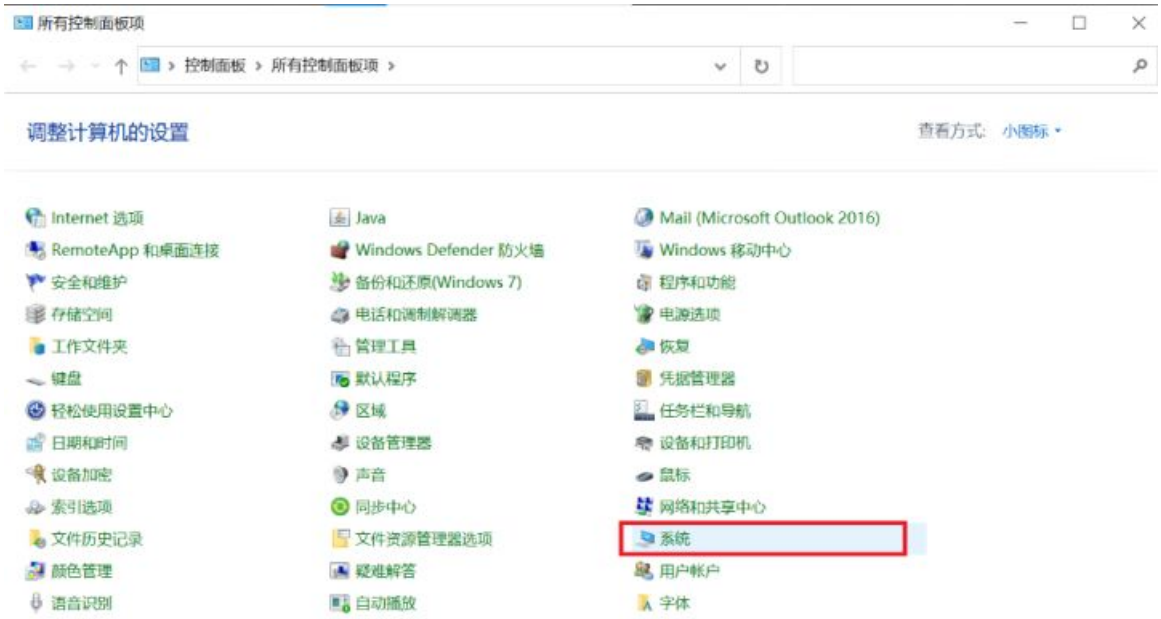
（2）设置环境变量

运行Java程序时需要进行两个步骤：第一步编译，将源文件编译成字节码，对应的是javac命令；第二步解释，解释执行平台无关的字节码程序，对应的是java命令。

虽然计算机安装JDK并且JDK的安装路径下面包含了这两个命令，但是计算机并不知道到哪里找这两个命令，在cmd执行这两个命令的时候，可能会提示“不是内部或外部命令，也不是可运行程序或批处理文件”。因此，设置环境变量，就是要告诉计算机这些命令所处的路径，以方便计算机找到它们。

本例中，由于安装的是JDK 1.8，安装程序会自动设置环境变量，因此手动设置环境变量这一步骤可以省略。如果安装的是低版本的JDK（比如1.5、1.6等），手动设置环境变量则是必需的步骤。具体设置方法为：

点击“控制面板”->“系统”->“高级系统设置”：



编辑切换为居中

添加图片注释，不超过 140 字（可选）

编辑切换为居中

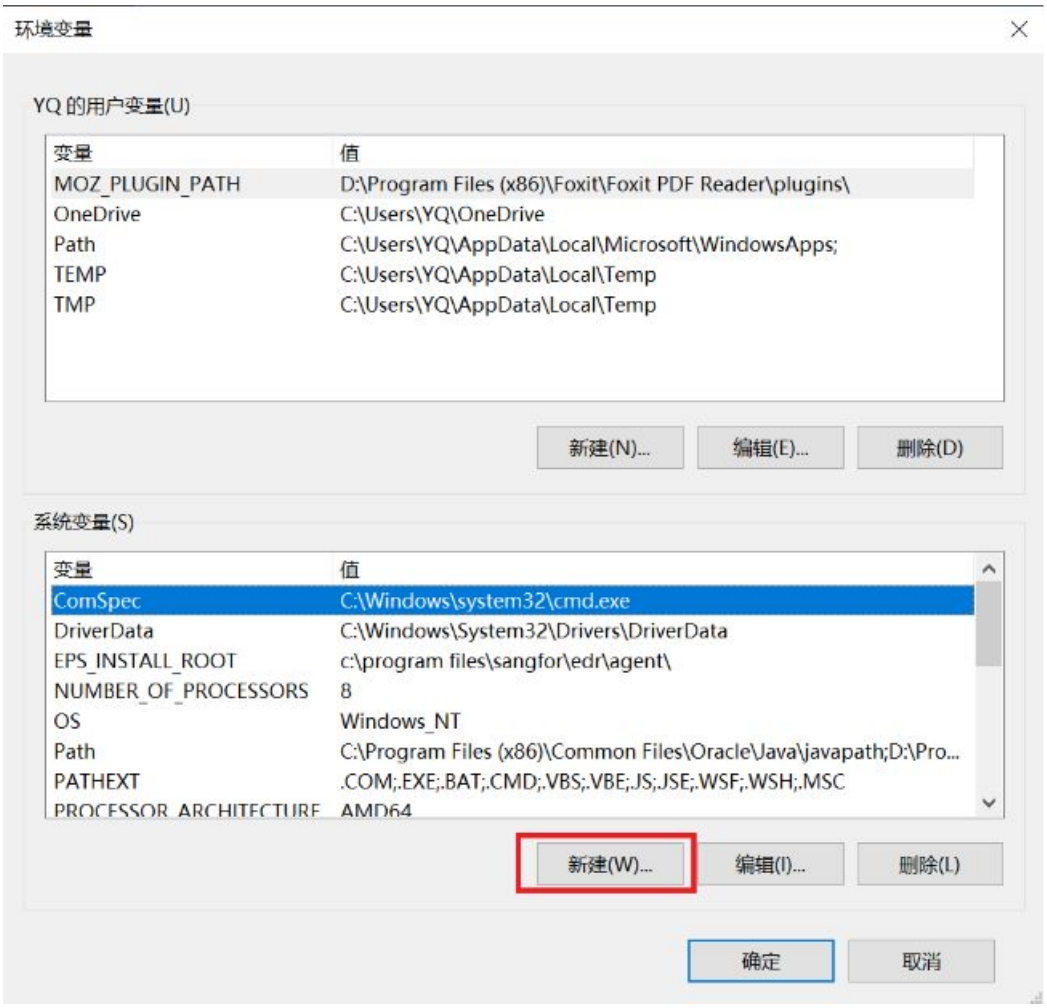
添加图片注释，不超过 140 字（可选）

弹出如下的“系统属性”对话框时，在“高级”标签页下点击下方的“环境变量”按钮：

编辑

添加图片注释，不超过 140 字（可选）

在“系统变量”模块下点击“新建”按钮：



编辑切换为居中

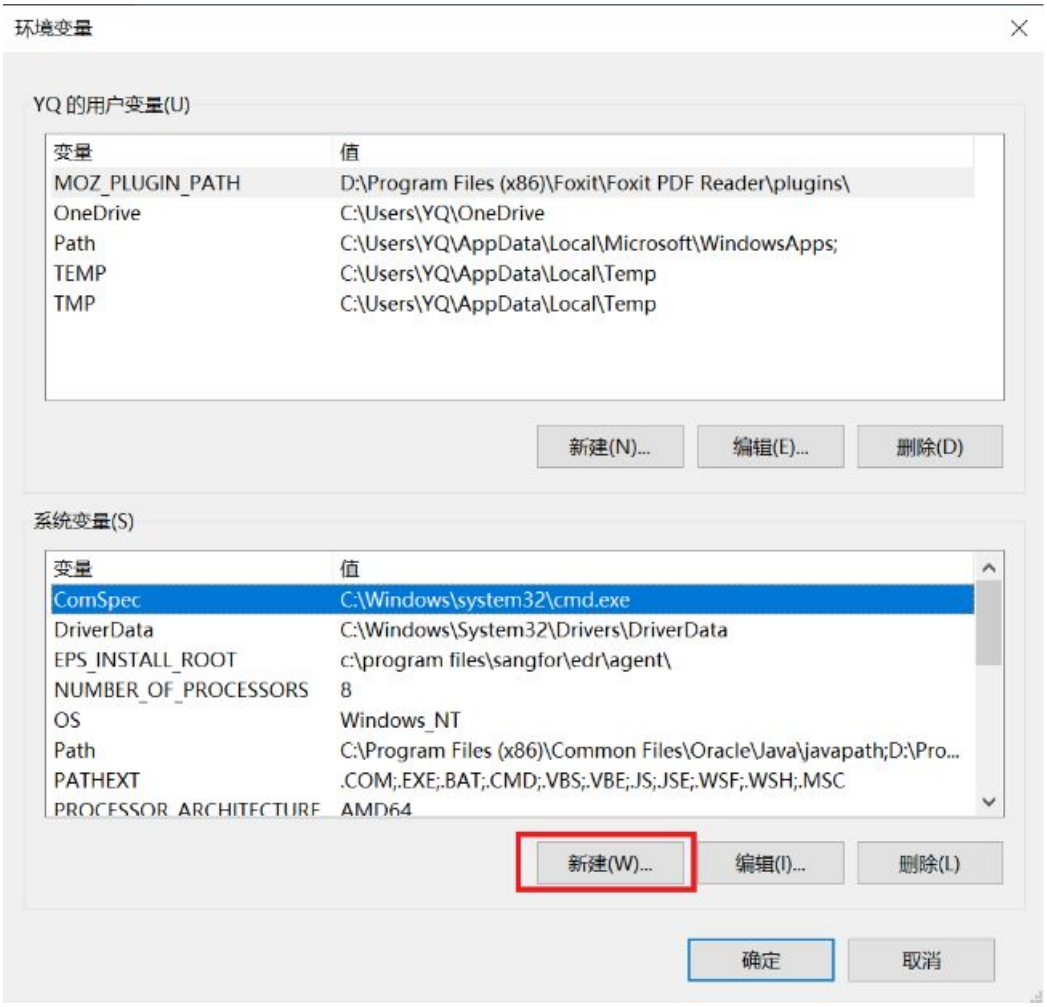
添加图片注释，不超过 140 字（可选）

新建两个变量：

变量名称为“JAVA_HOME”，变量值为“C:\Program Files\Java\jdk1.8.0_191”；

变量名称为“CLASSPATH”，变量值为“.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar”（注意前面有一个“.”和“;”）。

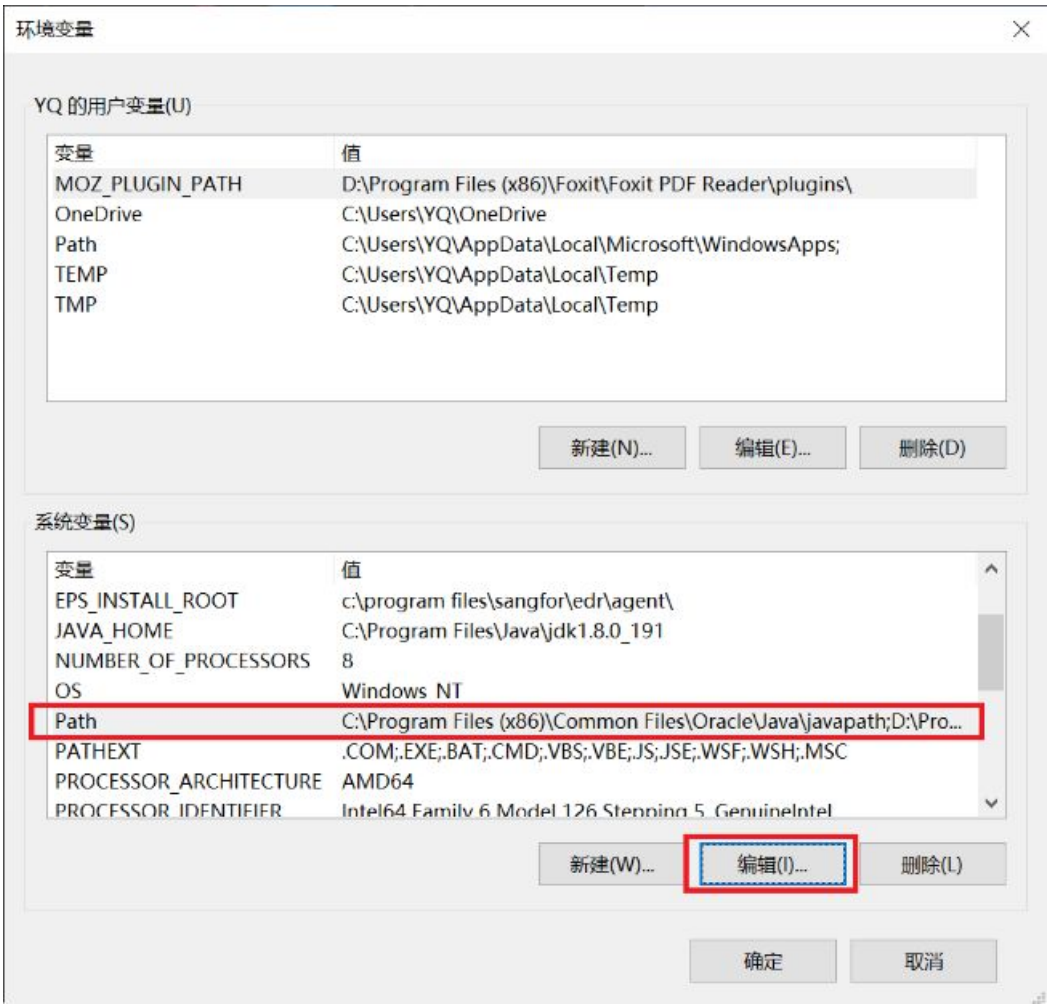
如图所示：



编辑切换为居中

添加图片注释，不超过 140 字（可选）

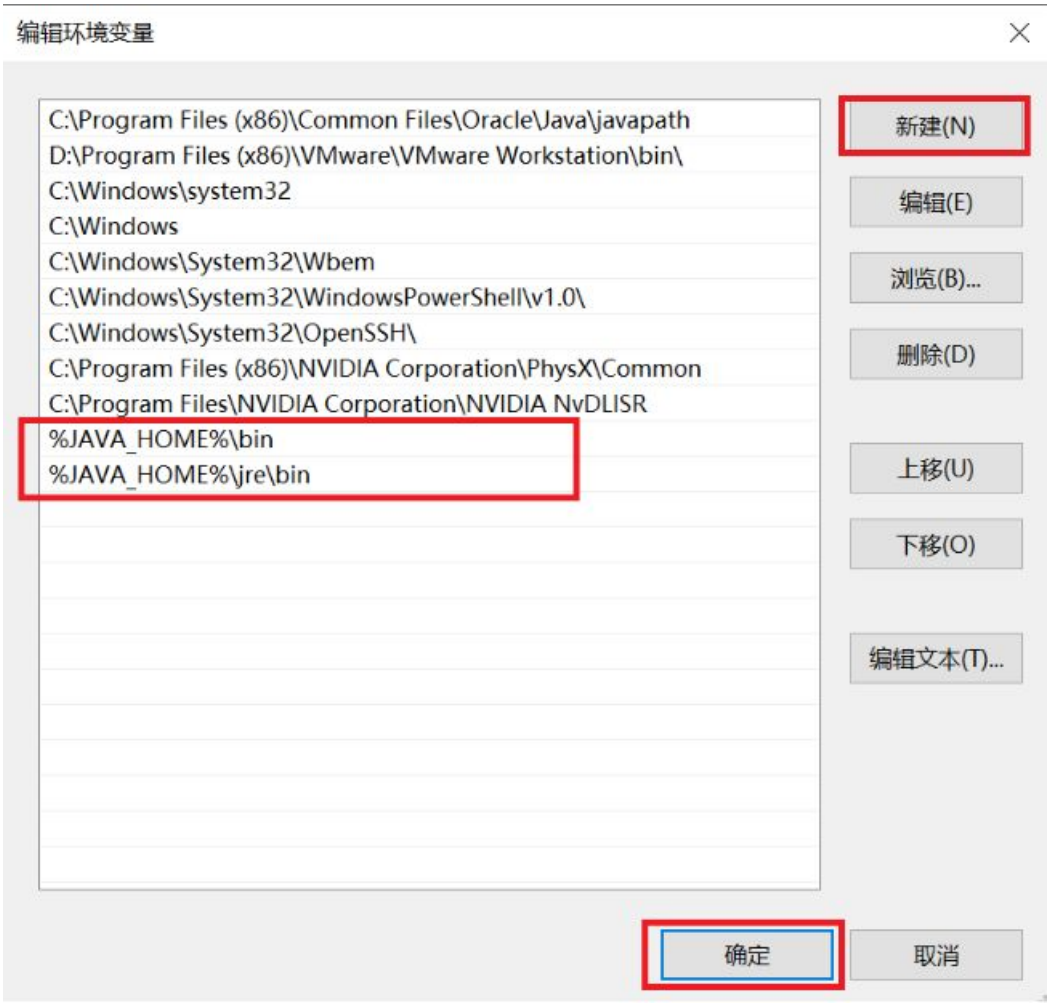
继续在“系统变量”列表中找到并选中“PATH”变量，点击“编辑”按钮：



编辑切换为居中

添加图片注释，不超过 140 字（可选）

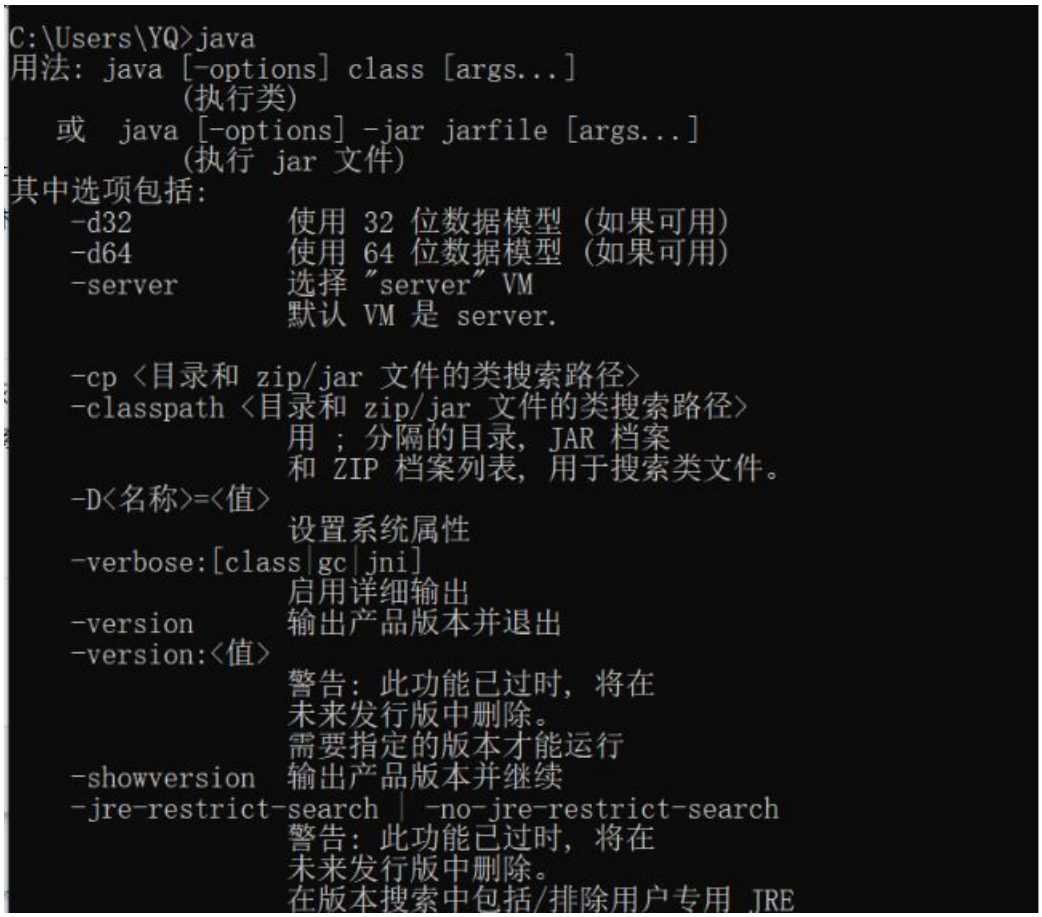
继续点击“新建”按钮，增加2条环境变量“%JAVA_HOME%\bin”和“%JAVA_HOME%\jre\bin”，完成后点击“确定”，如图所示：



编辑切换为居中

添加图片注释，不超过 140 字（可选）

运行cmd，输入命令“java”，出现一连串的指令提示，说明环境变量配置成功了：

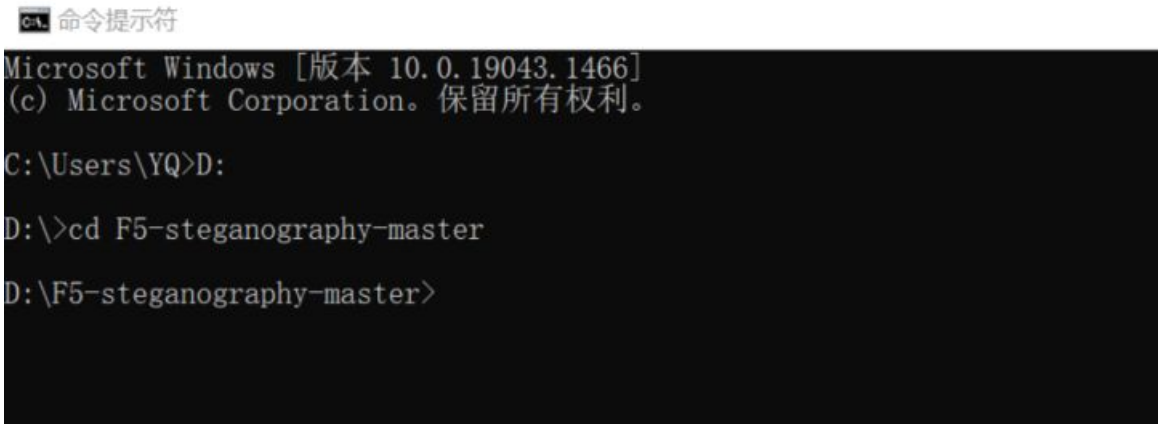


编辑切换为居中

添加图片注释，不超过 140 字（可选）

三、测试F5隐写工具软件的演示程序（demo）

在cmd中，依次输入命令“D:”和“cd F5-steganography-master”进入F5隐写工具软件所在路径：



编辑切换为居中

添加图片注释，不超过 140 字（可选）

继续输入命令“e.bat”直接运行e.bat程序：

```
D:\F5-steganography-master>e.bat

D:\F5-steganography-master>del lopez.jpg
找不到 D:\F5-steganography-master\lopez.jpg

D:\F5-steganography-master>java Embed lopez.bmp lopez.jpg -c "" -e bin.noise -p pleasechangethispassphrasetoyourown
DCT/quantisation starts
315 x 199
got 99840 DCT AC/DC coefficients
one=7167
large=7154
expected capacity: 10665 bits
expected capacity with
default code: 1333 bytes (efficiency: 1.4 bits per change)
(1, 3, 2) code: 888 bytes (efficiency: 1.7 bits per change)
(1, 7, 3) code: 570 bytes (efficiency: 2.1 bits per change)
(1, 15, 4) code: 354 bytes (efficiency: 2.5 bits per change)
(1, 31, 5) code: 213 bytes (efficiency: 3.0 bits per change)
(1, 63, 6) code: 126 bytes (efficiency: 3.6 bits per change)
(1, 127, 7) code: 63 bytes (efficiency: 3.6 bits per change)
Permutation starts
Embedding of 1736 bits (213+4 bytes) using (1, 15, 4) code
730 coefficients changed (efficiency: 2.3 bits per change)
357 coefficients thrown (zeroed)
1736 bits (217 bytes) embedded
Starting Huffman Encoding.

D:\F5-steganography-master>_
```

编辑切换为居中

添加图片注释，不超过 140 字（可选）

执行e.bat程序后的效果为：bin.noise文件中的信息被嵌入到lopez.bmp图像文件中，并生成了lopez.jpg文件。此时，若用户同时打开lopez.bmp和lopez.jpg进行比较，是很难发现两者的不同点的：

编辑

lopez.bmp

编辑

lopez.jpg

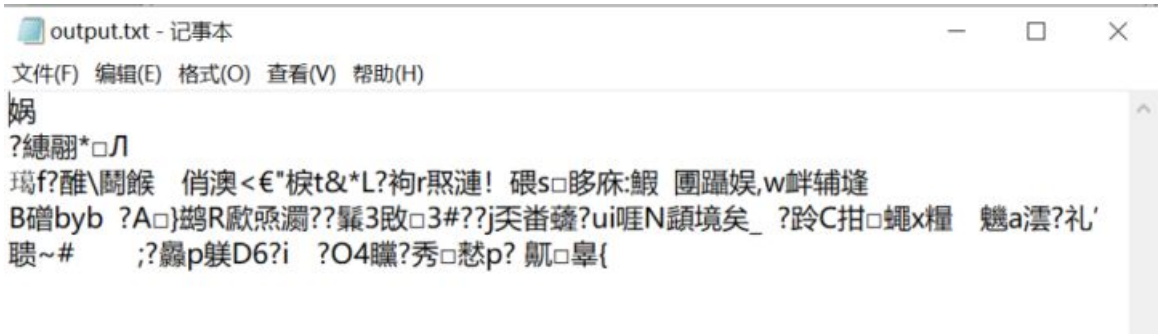
继续在cmd中输入“d.bat”，直接运行d.bat程序：

```
D:\F5-steganography-master>d.bat

D:\F5-steganography-master>java Extract lopez.jpg -p pleasechangethispassphrasetoyourown
Huffman decoding starts
Permutation starts
99840 indices shuffled
Extraction starts
Length of embedded file: 213 bytes
(1, 15, 4) code used

D:\F5-steganography-master>
```

编辑切换为居中

添加图片注释，不超过 140 字（可选）

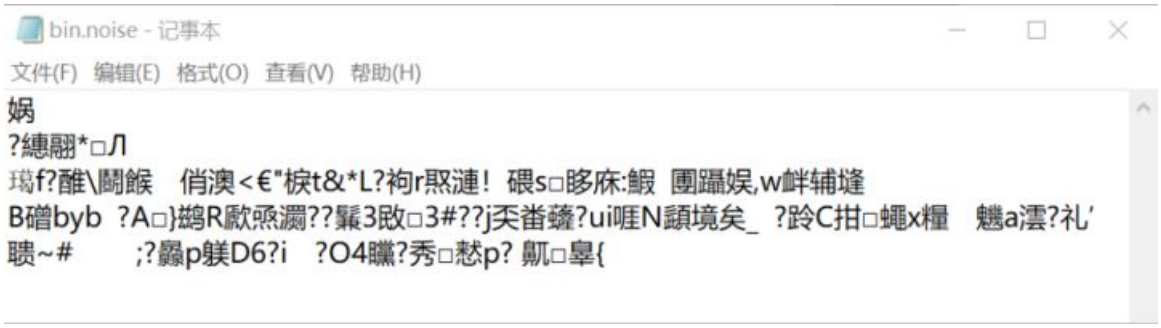
执行d.bat程序后的效果为：从lopez.jpg文件中提取到了隐藏信息，并被保存至output.txt文件中。此时，用户可同时打开output.txt和bin.noise进行比较，发现两者的内容是相同的：



编辑切换为居中

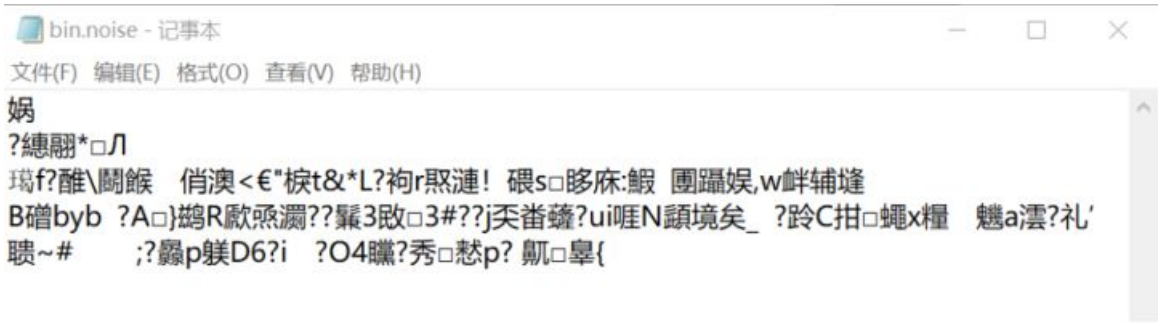
添加图片注释，不超过 140 字（可选）

output.txt的内容



编辑切换为居中

添加图片注释，不超过 140 字（可选）



编辑切换为居中

bin.noise的内容

四、运行Java命令隐藏和提取自定义信息

通过运行演示程序我们发现，隐藏信息时所使用的命令为：

java Embed lopez.bmp lopez.jpg -c "" -e bin.noise -p pleasechangethispassphrasetoyourown

其中，“lopez.bmp”（原载体文件）、“lopez.jpg”（嵌入隐藏信息后的载体文件）、“bin.noise”（信息文件）以及“pleasechangethispassphrasetoyourown”（程序的执行口令）等参数均可根据需求进行设置或修改。

提取隐藏信息时所使用的命令为：

java Extract lopez.jpg -p pleasechangethispassphrasetoyourown

其中，“lopez.jpg”（嵌入隐藏信息后的载体文件）以及“pleasechangethispassphrasetoyourown”（程序的执行口令）等参数均可根据需求进行设置或修改。

弄清了命令和参数后，我首先自建了一张图片1.bmp和一个保存了自定义信息的文件1.noise：

编辑切换为居中

1.bmp的内容

1.noise的内容

尝试使用1.bmp图像文件隐藏1.noise中的信息，并将隐藏信息提取到output.txt中，程序执行口令设置为“123456”。这个过程的操作步骤如下：

（1）在cmd中输入“java Embed 1.bmp 1.jpg -c "" -e 1.noise -p 123456”，将1.noise中的信息嵌入到1.bmp，并生成1.jpg文件：

```
D:\F5-steganography-master>java Embed 1.bmp 1.jpg -c "" -e 1.noise -p 123456
DCT/quantisation starts
1024 x 714
got 1105920 DCT AC/DC coefficients
one=3537
large=178342
expected capacity: 180075 bits
expected capacity with
default code: 22509 bytes (efficiency: 1.9 bits per change)
(1, 3, 2) code: 15006 bytes (efficiency: 2.6 bits per change)
(1, 7, 3) code: 9646 bytes (efficiency: 3.3 bits per change)
(1, 15, 4) code: 6001 bytes (efficiency: 4.1 bits per change)
(1, 31, 5) code: 3627 bytes (efficiency: 5.0 bits per change)
(1, 63, 6) code: 2142 bytes (efficiency: 5.9 bits per change)
(1, 127, 7) code: 1238 bytes (efficiency: 6.9 bits per change)
Permutation starts
Embedding of 496 bits (58+4 bytes) using (1, 127, 7) code
86 coefficients changed (efficiency: 5.7 bits per change)
3 coefficients thrown (zeroed)
496 bits (62 bytes) embedded
Starting Huffman Encoding.

D:\F5-steganography-master>
```

编辑切换为居中

添加图片注释，不超过 140 字（可选）

打开生成的1.jpg图片，发现其内容几乎与1.bmp一模一样：

编辑切换为居中

添加图片注释，不超过 140 字（可选）

1.jpg的内容

（2）在命令提示符中输入“java Extract 1.jpg -p 123456”, 将1.jpg中的隐藏信息提取到output.txt文件中：

```
D:\F5-steganography-master>java Extract 1.jpg -p 123456
Huffman decoding starts
Permutation starts
1105920 indices shuffled
Extraction starts
Length of embedded file: 58 bytes
(1, 127, 7) code used

D:\F5-steganography-master>
```

编辑切换为居中

添加图片注释，不超过 140 字（可选）

执行完毕后，比较output.txt和1.noise的内容，发现两者一致，隐藏信息提取正确：

编辑切换为居中

output.txt的内容

Bingo！

作者：袁泉，深信服安全服务认证专家，产业教育中心资深讲师，曾任职于国防科技大学信息通信学院，从事计算机网络、信息安全专业教学和科研工作十余年，持有HCNA（SECURITY）和HCNA （R&S）证书。熟悉TCP/IP 协议及网络安全防护体系架构，具有丰富的计算机网络管理、运维与安防实践经验。