Mac Charles v4.0.2 detailed crack tutorial

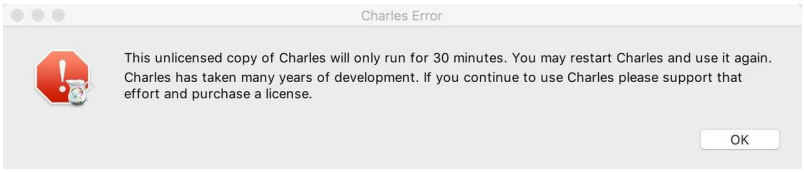
Author wooyoo (/u/7c72e30bd146) + attention
2016.12.18 23:01 Word Count 2608 Read 298 Comments 0 Like 2
(/u/7c72e30bd146)

background

Charles is a very good capturing software, especially in the mac operating system. Charles is a commercial software, although the experience version can use all the features, but there are several inconvenient use of the following:

1. Start with 10s prompts to buy the window, and every 4 minutes, when you click on Charles to operate again, it will pop up a 5s prompt for the purchase window. Tips to buy window screenshots are as follows:

2. Once run for more than 30 minutes, it will pop up the following error prompt window, and terminate the program:



For the sake of **study and research** , after successfully breaking Charles, Charles can be found as a very typical Jar package to solve the teaching, because the crack process has the following characteristics:

1. Jar has been confused by code.
2. Crack more ideas.
3. Can be used more than the crack tools.

Before explaining Charles in detail, let me introduce the environment:

- MacOS 10.11.6

- Charles v4.0.2
- Jdk 1.8

Well, let's take a closer look at how to break Charles.

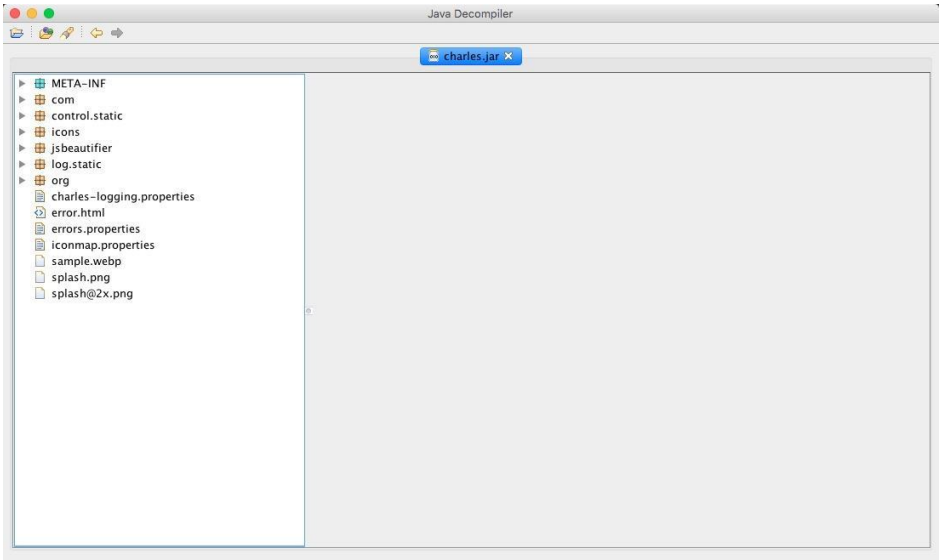
Crack the process

Thinking one

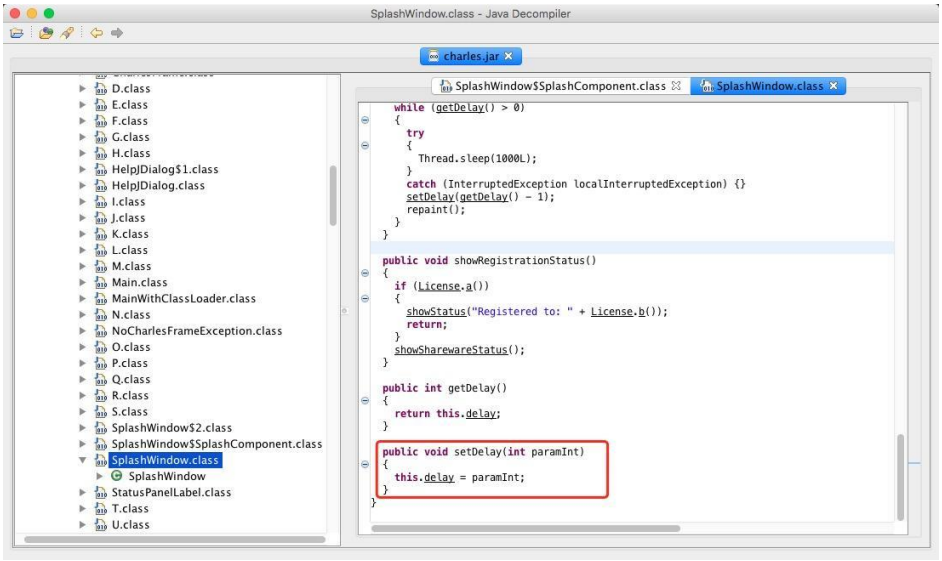
Code analysis and positioning

In the background we introduced, Charles experience version did not function castration, so the first crack idea is to remove Charles on the experience version of the restrictions made. Anyway, the first step in all Jar packages is to read the code through decompile software. Here we use the JD-GUI, which is a very good java anti-compiler software, in addition to reading the anti-compiled code, you can also search.

We first enter Charles, mac can be right through the Charles icon, and then select the display package content. Go to the Java directory and open the charles.jar with the JD-GUI:



After the observation, we found that charles.jar after the code confusion, so that many of the logic within the program can only rely on experience to guess. Since there is a "Delay" prompting the purchase window, we first search for all places where the "Delay" string has appeared. By analyzing the search results, we boldly guess that the SplashWindow is used to handle the relevant logic of the window, and the setDelay method is used to set the window's display time:



Then we searched all the code that called the setDelay method and was fortunate that we suddenly found the code for setting the start interface 10s delay time and setting the 5s delay time after every 4 minutes.

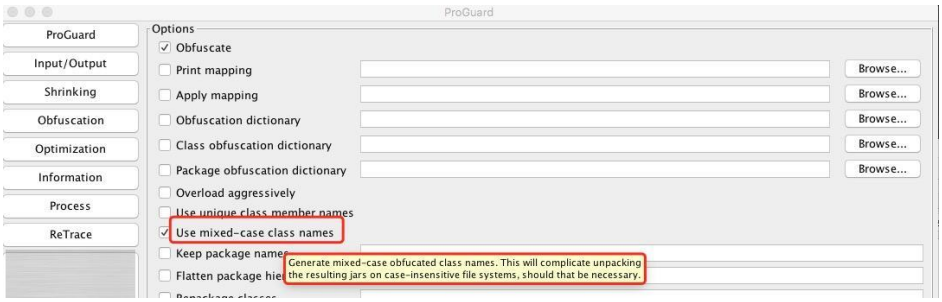
Delete the 10s delay of the startup interface

The first is in the V. class at com.xk72.charles.gui:

```
final class V
  implements Runnable
{
  V(Main paramMain, SplashWindow paramSplashWindow) {}

  public final void run()
  {
    this.a.showSharewareStatus(); // 显示软件注册信息
    this.a.setDelay(10); // 设置窗口维持10s钟之后消失
  }
}
```

In order to give us the next operation to lay the foundation, we first need to extract the jar package. But because Charles has done a class name-based code confusion, this will cause the jar package to fail to extract the file name on a case-insensitive operating system (such as a.class will be in the process of decompression by A.class to cover). This is a very common confusion technology, because we are the most commonly used windows and mac operating system is not distinguish between the file name case operating system, so this confusion can be most of the bad boy to interception. The following figure shows a confusing screenshot of the ProGuard confusing tool:



In order to solve this situation, I generally use the Linux system to extract the class name through the confused Jar package. After decompression, we use the JBE bytecode modification tool to modify the V.class bytecode. For the use of JBE, we can

refer to my other article: http://www.jianshu.com/p/a61f0f44705b (http://www.jianshu.com/p/a61f0f44705b) . Modify V.class method is very simple, only need to run the corresponding byte code to delete, leaving only return can be.

Delete 5s delay every 4 minutes

The second is located in SplashWindow's a method:

```
public static void a(Window paramWindow, Runnable paramRunnable)
{
  (paramWindow = new SplashWindow(paramWindow)).showSharewareStatus();
  paramWindow.setDelay(5); // Delay 5s
  paramWindow.setVisible(true);
  a.a(new aa(paramWindow, paramRunnable));
}
```

Modify the method is very simple, just the corresponding byte code in addition to return all deleted, make it an empty method can be.

Remove the code for half an hour to exit

Finally, we need to delete the code for half an hour to exit. Positioning code is very simple idea, search error dialog box can appear in the string, where we search through the JD-GUI "unlicensed copy", from the search results, we can find com.xk72.charles package e. There is a corresponding logic in the class:

```
final class e
  implements Runnable
{
  e(d paramd) {}

  public final void run()
  {
    k.b();
    this.a.a.error("This unlicensed copy of Charles will onl
    this.a.a.exit(0, true);
  }
}
```

弹出错误对话框

退出程序

Modify the same method, the use of JBE run method corresponding to the byte code in addition to return all deleted, so that it can be an empty method.

At last

Will all of the above hacked class file into the original jar package (must be in the linux system), replace the original charles app under the charles.jar, we found that after the start has been successfully cracked.

Thinking two

Code analysis and positioning

In fact, thinking is not a conventional idea, in most cases, we will deal with "License" method to crack. If you go through a complete way of thinking a way, then it will certainly find com.xk72.charles package is called a license.class class, and the class a method is often called by external, used to determine whether it has been registered Success, such as every 4 minutes to show the logic of the prompt window:

```
public final void d(Object paramObject)
{
  if (this.e != null)
  {
    this.f = this.e;
    c(null);
  }
  if ((!License.a()) && (System.currentTimeMillis() - d > 240320L))
  {
    if (b == null)
    {
      b = paramObject;
      c = this;
      SplashWindow.a(KeyboardFocusManager.getCurrentKeyboardFocusManager().getActiveWindow(), new g(this));
      return;
    }
    b = paramObject;
    c = this;
    return;
  }
  c(paramObject);
  b(paramObject);
}
```

如果软件未注册 并且过了约等于4分钟的时间

显示SplashWindow

Modify to determine whether the method has been registered

We can boldly identify the method is used to determine whether it has been registered. Follow this idea, we first try to change the method to always return true. Through the JBE will be a method corresponding to the bytecode modified to:

```
iconst_1
ireturn
```

Then start Chalres found that the card has been stuck in the Loading Preferences interface. In order to find out the reason, and then introduce a **small skill** , we can start the program through the command line : `java -jar ../Charles.app/Contents/Java/charles.jar` , and then we can see the console will output the corresponding error stack. From the error stack we can see the empty pointer exception:

```
Caused by: java.lang.NullPointerException
        at com.xk72.charles.License.b(Unknown Source)
        at com.xk72.charles.gui.SplashWindow.showRegistrationStatus(Unknown Source)
        at com.xk72.charles.gui.U.run(Unknown Source)
```

You can see the license of the b method:

```
public static String b()
{
  License localLicense = b;
  switch (p.a[localLicense.f.ordinal()])
  {
  case 1:
    return localLicense.d;
  case 2:
    return localLicense.d + " - Site License";
  case 3:
    return localLicense.d + " - Multi-Site License";
  }
  return localLicense.d;
}
```

Modify the display license name method

We boldly guess that the method is used to display the license name, we will modify it to return to a fixed string (what would you like to call what). If the reader is not very familiar with the java byte code, there is one of the most simple approach is to write a demo java program, and then use JBE to see the corresponding bytecode:

```
ldc "wooyoo"
areturn
```

Continue to run again VerifyError:

This error is caused by a failure of the JVM bytecode verification. The problem is very high probability because of JBE bug, in order to solve this problem, we have the following three solutions:

1. Turn off JVM bytecode checking.
2. Use javassist to modify bytecode.
3. Compile your own license

Turn off JVM bytecode checking

In order to close the JVM bytecode validation, we need to add additional JVM startup parameters. Because my computer's java environment is 1.8, so only by adding-noverify parameters, if it is 1.7, you can also use-XX: -UseSplitVerifier. We can in the Charles program directory info.plist file inside to add off parameters:

```
......
<key>JVMOptions</key>
<array>
<string>-noverify</string>
......
```

Use javassist to modify bytecode

Javassist is often used to dynamically modify the bytecode, widely used in the various java framework inside (where it is a bit to use to crack the software is really a bit embarrassed). Modify the code as follows:

```
public class Test {
    public static void main(String[] args) throws NotFoundException, CannotCompileEx
        ClassPool classPool = ClassPool.getDefault();
        classPool.insertClassPath("/path/to/charles.jar");
        CtClass license = classPool.get("com.xk72.charles.License");
        // 修改a方法
        CtMethod aOldMethod = license.getDeclaredMethod("a", null);
        CtMethod aNewMethod = CtNewMethod.copy(aOldMethod, license, null);
        aOldMethod.setName("a_old");
        aNewMethod.setBody("{return true;}");
        aNewMethod.setName("a");
        license.addMethod(aNewMethod);
        // 修改b方法
        CtMethod bOldMethod = license.getDeclaredMethod("b", null);
        CtMethod bNewMethod = CtNewMethod.copy(bOldMethod, license, null);
        bOldMethod.setName("b_old");
        bNewMethod.setBody("{return \"wooyoo\";}");
        bNewMethod.setName("b");
        license.addMethod(bNewMethod);
        // 输出新的License.class
        license.writeFile("/path/to/License.class");
    }
}
```

It should be noted that, when using javassist, we can not remove the original method inside the logic. Based on this, we have the following two practices:

- First copy the original method, and then call the setBody method into the new method into its own logic. The above code is used in this way.

- You can call the insertBefore method in the original method, for example:

```
CtMethod bMethod = license.getDeclaredMethod("b", null);
bMethod.insertBefore("{if(1<2) return \"wooyoo\"}")
```

We will be through an implementation of the if statement, to achieve only the effect of their own logic.

Compile your own license

The method is also a reference to the network of another person's cracked article: http://blog.csdn.net/endlu/article/details/52175787 (http://blog.csdn.net/endlu/article/details/52175787) , I will not repeat here, interested readers can see this article.

At last

Will be the hack of the License.class file into the original jar package. Here we do not have to complete the package in the Linux system, you can use the jar command to complete the replacement of class files directly : `jar –uvf charles.jar com/xk72/charles/License.class` . And then replace the original jar package charles app under the charles.jar, we found that after the start has been successfully cracked.

to sum up

Again, the above tutorials are based on the purpose of learning and research, and we strongly recommend that you still have the ability to buy genuine software. Well, let's summarize the points of knowledge we need to master through this break:

- Jar can not be compressed or decompressed on an operating system that does not distinguish between file names. We can use the Linux operating system to complete the compression or decompression operations.
- Cracking ideas are generally modified to deal with "License" logic. But we can not be limited to this, for example, in this case, we can remove the experience version of the restrictions, the same to achieve the purpose of cracking.
- In some cases if the crack is unsuccessful, we can start the program through the command line, and then through the observation console output error stack to see where the problem is in the end. Also if the program has a log file, we can also log the file to analyze the cause of the error.
- We can JBE, javassist or even their own way to re-edit the original bytecode.

Reference materials

- JVM Opcode Reference (http://homepages.inf.ed.ac.uk/kwxm/JVM/index.html)
- JVM class File Format (http://docs.oracle.com/javase/specs/jvms/se8/html/jvms-4.html#jvms-4.10.1.9.ldc)
- Java bytecode operation tool (http://blog.csdn.net/hudashi/article/details/50884742)

wooyoo (/u/7c72e30bd146)
Wrote 17664 words, was 28 people concerned, won 58 like (/u/7c72e30bd146)
+ Concerned about

Back-end engineer @ billion Fangyun, personal blog address: www.kissyu.org