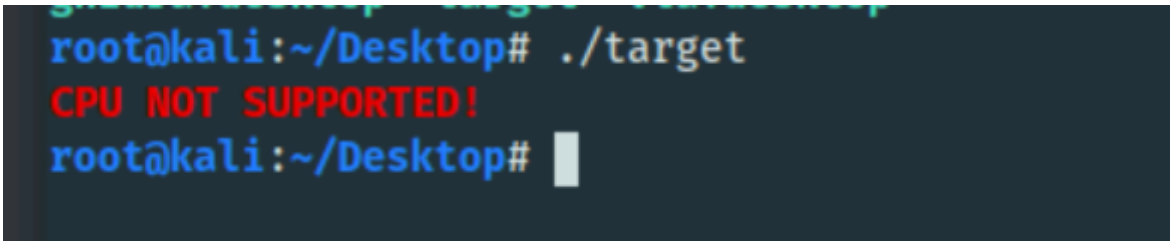
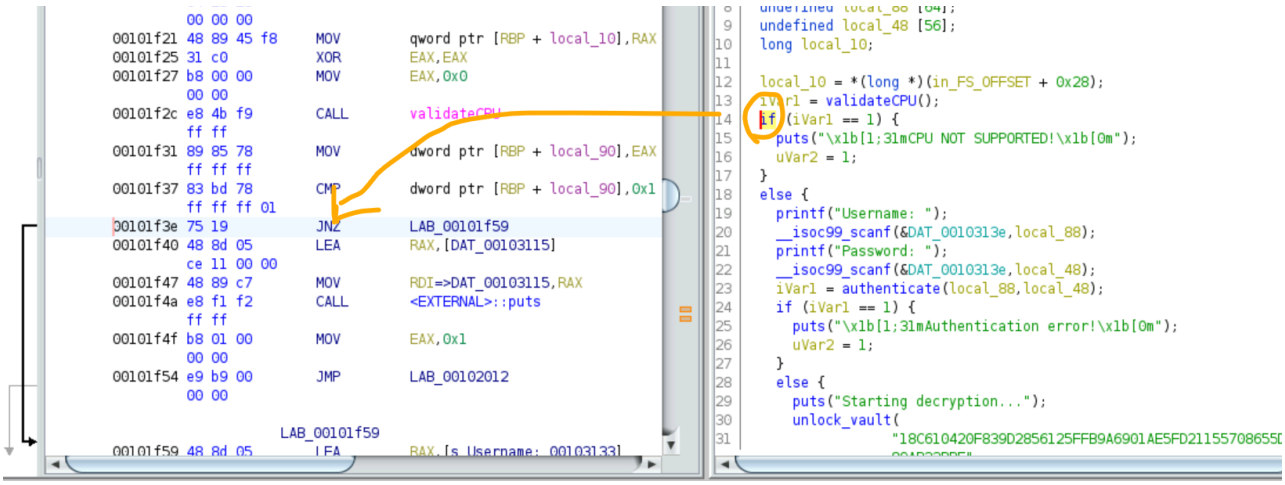
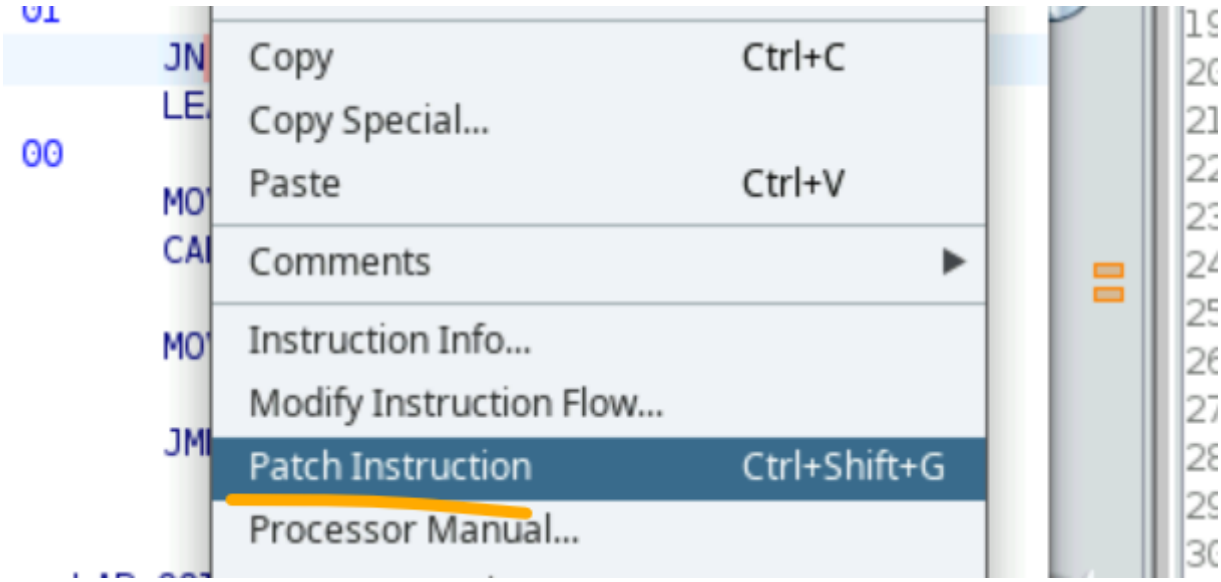
Patching

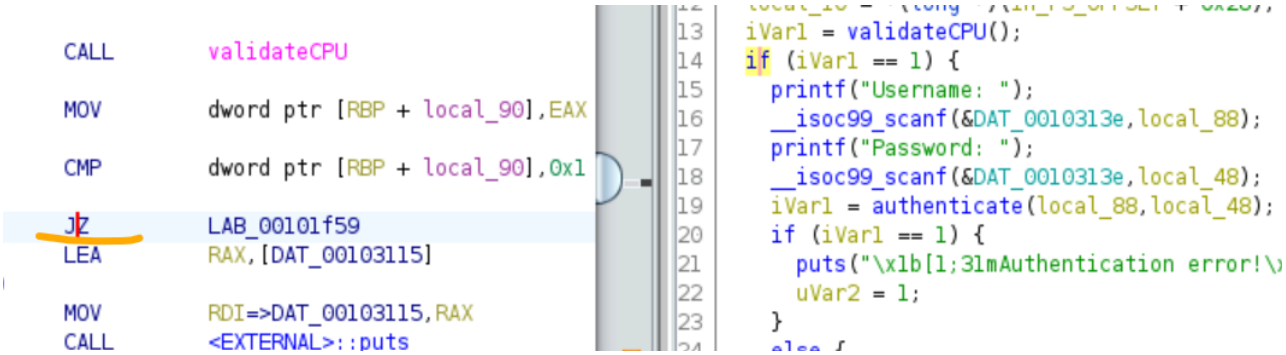
其实就是通过更改判断条件的汇编代码，使得改变程序的control flow，让它bypass各种判断限制。



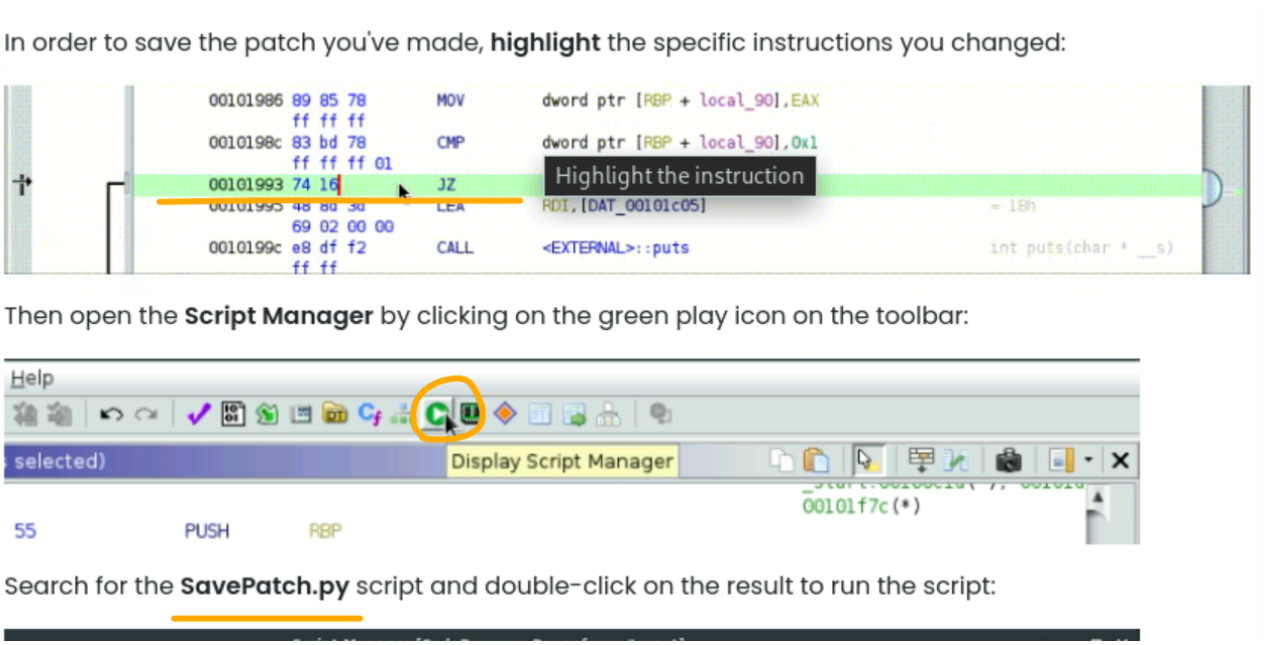


可以修改汇编码，比如JNZ 改为 JZ。





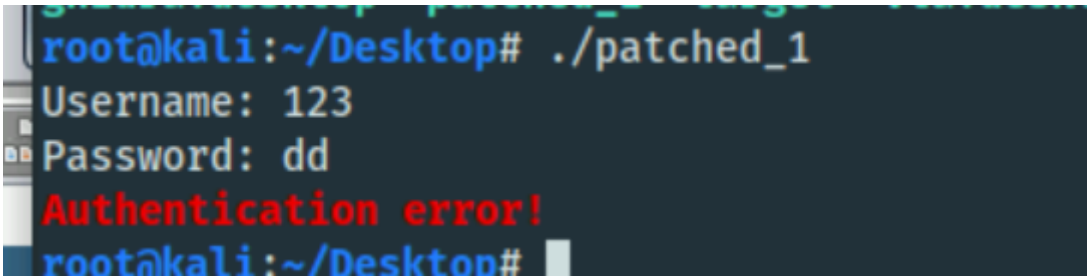
更改完后，highlight它，然后按script manager，找savePatch.py.



SavePatch 是一个plugin，允许你manually control the changes exported.

> **Note:** While Ghidra does allow exporting binaries from the **File > Export Program** menu, it not only saves the changed instructions but also any automatic changes made when the project was originally imported and analyzed, meaning you are likely to get a non-working binary as a result. Therefore, it is better to use the **SavePatch** plugin which allows you to manually control the changes exported. This plugin has already been installed for you.

这个时候允许你修改的文件，就不是CPU Warning了。



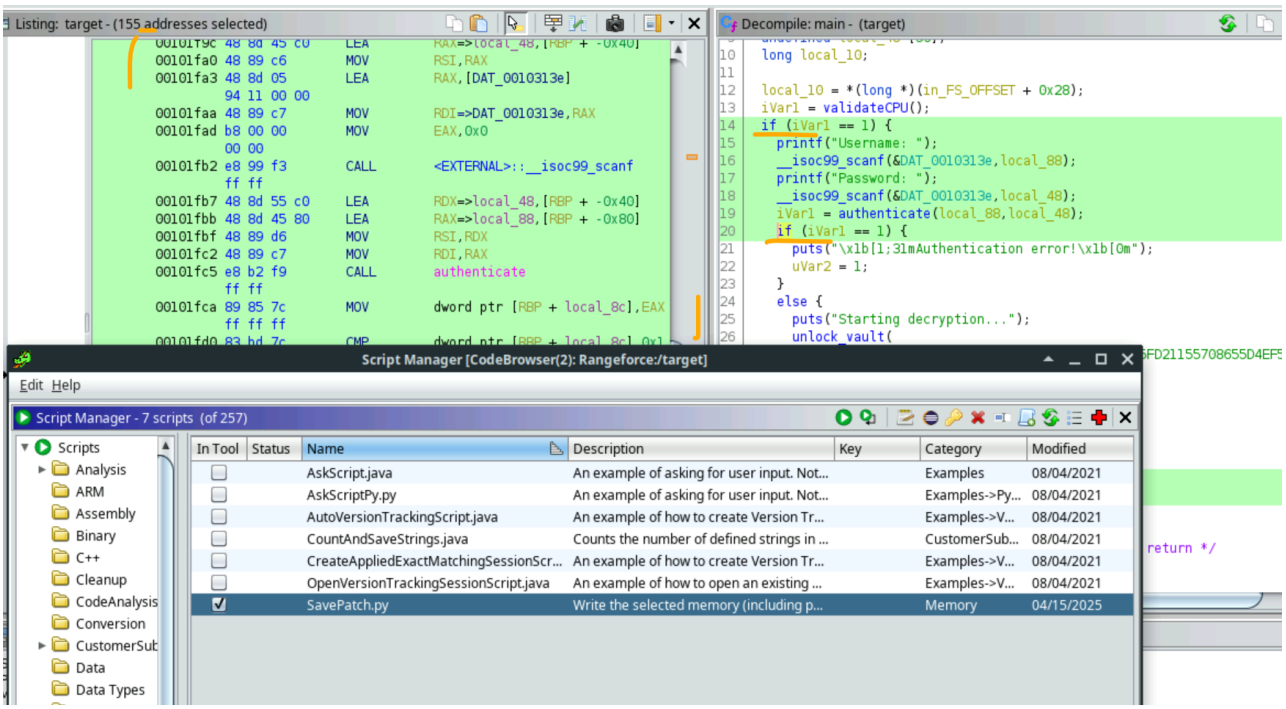
第二个patch

highlight第二个patch.

Due to the way the **SavePatch** script works, it will not remember any previous patches. So, only selecting this change is not enough. This would leave the previous change unsaved (though the authentication check would be bypassed, the CPU check would not be bypassed). You could import the previously patched binary into Ghidra and analyze and patch it then, but this approach can be too time-consuming.

Instead, you can **highlight both of your patches** at once and then use the **SavePatch** script to apply the changes. You have to highlight the whole memory range between the patches — just highlighting the changed lines individually won't work. You can use the black markers on the side to find the changes you've made.

After you have selected both instructions, run the **SavePatch** script and save the binary to `/root/Desktop/patched_2`.



选完savePatch后，按右上角的开始按钮，保存Patch。