For this challenge was given no description/text inside the usual Google Form, so nothing to report here. Inide the form we notice there is a PIN input to give. We’re only given the .apk file, for us ready to inspect.

Solution

Let’s start from the usual:

*jadx -d out goingseriousnative.apk*

From the path “out/sources/com/mobiotsec/goingseriousnative/MainActivity.java”, there are some interesting things to notice:

* there is a native function *checkFlag*
* there is a library which is being loaded, which is “goingseriousnative”
* the other parts of the code simply set a text being changed, before, currently and after, the on click we get a string which gets set in the main widget. The value of the flaf is in clear, but as said, a PIN is required and so we will delve into further analysis.

We can see the binary gets loaded and we will look into “out/resources/lib/x\_86\_64/libgoingseriousnative.so” with a tool like Ghidra.

In the list of functions, we can notice “Java\_com\_mobiotsec\_goingnative\_MainActivity\_checkFlag” function, which uses two parts; *preprocessing* checking inputs and *validation*.

The checkflag itself does nothing, instead the *preprocessing* is much more interesting. We can see there’s a loop here, using a delimiter with the *strtok* function. This time IDA tells us more things; we can see there is an input going on, using the *sscanf* function with the %d frmat, which convert an input string called *s* into the decimal format. Before the *strtok* function, there is the increment of a counter (ecx value, so var\_44) and the result of such elaboration inside the *var\_20* register.

We can also see an instruction which applies a shift over the current counter value, multiplying it by 4 given it is an integer. The result of such computation is hence saved into an array of integers, given a correct validation or -1.

The validate function gets called in the middle of preprocessing and actually takes back the computation from such function; we can see there is a compare between 5 and the actual value; it then jumps towards where the validate will receive the array of integers as argument and applies the delimiter character with the string. The Rbp+var\_14 is the loop counter, while the *Rbp+var\_10* is the array of integers which is received from *rdi* register as parameter.

The compare actually checks if the strings is made by 5 tokens; if decompiled in Ghidra, we see the loop iterates 5 times, one for each digit. There is also the compare with the value *64h* (100 in decimal representation). The *jge* assembly instruction makes us understand there is a compare between that value and the sum of such, which creates a 5-digit PIN which sum should be greater than *100*. If it less than that value, it returns (Ghidra shows this).