Flare-On 10 Challenge 1: X

By Nick Harbour (@nickharbour)

Overview

The X challenge is distributed as a package of many binaries. The challenge prompt instructed you to focus on the file X.exe, the primary game binary. All other binaries are present as framework requirements for this MonoGame program. This game was tested on Windows, but it will likely run and operate successfully on many forms of Linux and MacOS.

When you run the file X.exe you are presented with the following game screen.

Figure 1: X Game screen

The game only has 5 interactive buttons. A pair of up and down buttons to change the numeric values for the two digits of the code, and a lock button. The idea of this challenge is to enter the correct unlock code and press the lock button to “unlock” the gadget. There is a serious security flaw here, in that two digits is very easy to brute force. Flare-On Challenge #1’s are usually there to help us determine how many people are actually playing the game. In this respect, it is like a glorified captcha.

Solution 1: Brute Force

If you keep adjusting the two digits up from 00 towards 99 and hit the lock button each time, you will eventually hit 42. When 42 is set as the code the X will turn Green instead of Red and a pop-up window displaying the final Flag will appear, as shown below in Figure 2.

Figure 2: Victory Screen

Solution 2: Strings

Due to MonoGame's construction, most of the implementation of the game and its strings are contained in the file X.dll instead of X.exe itself. If you examine the strings of this file you will see a valid email address in the `@flare-on.com` domain: **`glorified_captcha@flare-on.com`**

You can automatically search for flare-on.com strings from the command line with the following command:

```
strings *.* | grep flare-on.com
```

Solution 3: Reverse Engineering The Program

If you simply must do things the hard way, you can reverse engineer the program by loading X.exe into dotPeek. Browse X->X->monogame1->Game1 to view a full decompilation of the X game source code. The most relevant function to examine is the callback function for the click action on the lock button. This function is intuitively named `_lockButton_Click()` and is shown below in Figure 3.

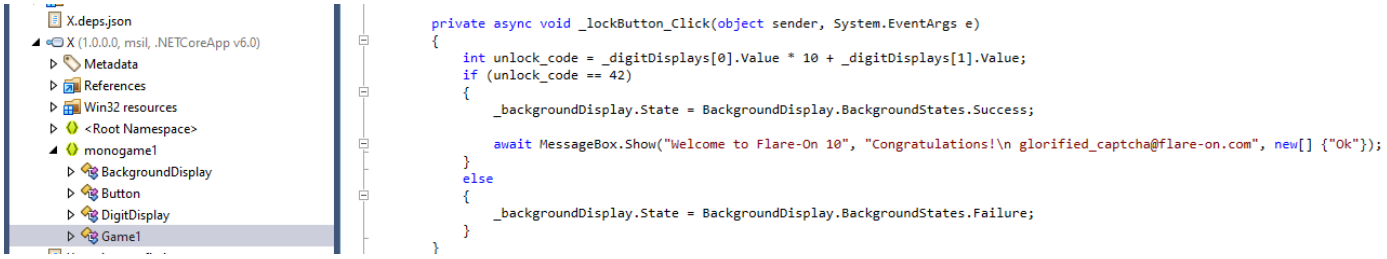

Figure 3: dotPeek Decompilation of _lockButton_Click()

The logic of this function is that it takes the two individual digit values and creates a single number from them. The leftmost digit is multiplied by 10 and added to the right digit. So if you entered '7' and '5' as the digits, this would produce the integer value 75, for example. It then checks if this combined value is 42. If it is 42, then it displays a MessageBox which displays the flag. If it is not 42, then it sets the background state to Failure, which results in the X turning red.