**Changes from original design**

I actually didn't have any changes from the original design. I started with a very simple idea and basically just added from there. So I have a few things that I was not thinking of including in the original game like the binary counter and the particle systems.

**Requirements**

**Media**

**Sound**: it has both background music and sound effects that were not used in class

**Images**: The only one I am using is the sprite that I made which is consistent with the theme and is a PNG

**Text**: all texts/fonts are appropriate and easy to read

**Canvas**: I am using canvas to draw animations, I am canceling them with window.cancelAnimationFrame() and they are all visually engaging and fit into the theme

**Interaction**

**Control**: the player types to increase the number of "hacks" they have and the type of upgrades they have. Depending on what upgrades and the amount of "hacks" the sprites change size and increase in number. The player must choose to either go a slow and safe route to hack and increase security or a fast and dangerous rout to lower security and hack faster.

**Usability Requirements**

the game pauses with window.onblur and unpauses with window.onfocus

**Teaching**: I want the player to figure stuff out on their own so essentially everything is locked until they start mashing buttons, when they do they will figure out that they "hack" by hitting random buttons. Some upgrades are then revealed, which teaches them what to do with all the points. As time goes on more and more upgrades are unlocked. Eventually a red bar will start filling up and two new different colored upgrades show up. When the user clicks on them it makes the red bar go down teaching them that the bar is bad and shouldn't get all the way up because this is game over. After they increase their security another button will appear that lowers their security but increases their hacks per tick. I think I effectively teach the player without ever directly telling them anything.

**Feedback**: Current hacks are displayed at the top and total hacks are displayed in binary at the bottom

**Difficulty**: the counterhack always goes up at a constant rate but increasing security costs more and more each time creating a difficulty curve.

**mouse and onscreen UI**: there are buttons that buy upgrades that the user interacts with to buy upgrades

**Screens**: there are 3 screens, the main game, the game over, and the winning screen. I do not have a start screen because of the nature of incremental games I feel like it's better to just dump the player in and let them figure things out for themselves.

**Game Specific Requirements**

has a win lose condition, player wins if they get the final hack, they lose if they are counterhacked

player choices matter, they have to balance security with increase how fast theyre hacks increase

it has a little bit of depth, mostly just meant to be a feel good game watching numbers go up

its engaging because it is rewarding, the more you play the bigger numbers get, the bigger and more complex the particle system becomes and the amount of "code" on screen increases.

my sprites are randomized and being used in conjunction with the particle system

**Coding requirement**

everything is encapsulated in drawing APIs. All images are preloaded and I use object literals and the module patter. I have more than 2 function constructors that were created by me. I do repeat myself a lot with the code, but each button acts differently so I couldn't really put it in a loop. The code is commented, I'm using semi-colons, using naming conventions, and I got rid of my console.log calls.

**What went right and what went wrong**

The whole project went pretty well, I'm surprised how easily the binary was. The particles did give me a lot of trouble though. I do have one visual bug and I have no idea what is causing it, luckily it fits with the theme of my game.

**overall grade**

95% for me, I felt like I did a pretty good job, I went above and beyond for a few things like showing the total score in binary and I think it looks pretty good too.