Cyber Security Education

Design Document

By Liana Villafuerte, Bruce Wagner, Kyle Stead, and Hussein Okasha

Features and Testing inputs and outputs

Features	Expected Input/In Game Test	Expected Output	Actual Output
Movement	Inputting WASD and spacebar keys	W/up arrow moves up A/left arrow moves left S/down arrow moves down D/right arrow moves right Spacebar jumps P pulls up pause menu Esc quits game	W/up arrow moves up A/left arrow moves left S/down arrow moves down D/right arrow moves right Spacebar jumps P pulls up pause menu Esc quits game
Interaction controls	Shooting bullets with left mouse input	Left mouse click shoots bullets	Left mouse click shoots bullets
Map generating	A clear map generated with clear goals and what to do	Usable map with a path	Usable map with scrolling feature when falling with limited y-axis restraints
Turret generating	Turrets should be generated in certain parts of the map according to its generation	Turret in correct positions	Turret in correct positions
Turret projectiles	Turrets aim and shoot at player, causing them to die if hit	Aims and shoots at player and inflicts damage	Aims and shoots at player and inflicts damage
Damage inflicting and receiving	Bullet at enemy causes their death and vice versa for player	Any bullet causes death	Turret bullet causes player death and player bullet causes turret and core death

Features	Expected Input/In Game Test	Expected Output	Actual Output
Death reset	Everytime the player is hit by a bullet and dies, the player is forced to restart at spawn point as if from the beginning. Player also loses any level progress with cores destroyed.	Player gets reset to the very beginning at spawn point with 0 cores acquired upon death and turrets being respawned.	Player gets reset to the beginning spawn point with 0 cores acquired and turrets being respawned
Progression of levels	Everytime the player destroys all the cores in the level and supasses the vulnerability, the player then proceeds to the next level in the list of levels	Upon level completion, the game loads the map for the next level as well as the core positions and turret locations. It also generates the user's position at spawnpoint.	Player progresses to the next level with new cores, tiles, turrets, and spawn point locations.
Time score	User's time per level is stored into a database and the overall time for all the levels is stored at the very end.	Player's time scores are stored in a database for storage and later use within the game	Game stores the results in the scores.db file that is an sqlite3 database file with the user's input name and total time.
SQL injection vulnerability	The player is expected to input a certain string into the name input so as to overwrite the real time with a fake time	The game takes the fake time inserted by the player and adds it to the database and ignores the actual input time. Allowing the user to pass the first level.	The game takes the exploited time inserted by the player and adds it to the database and ignores the actual input time.
Cryptography (caesar cipher)	The player is expected to type in a caesar cipher string into the name so that they can essentially be invisible to the turrets	The game accepts the input and enables invisibility so that the player can bypass the level with ease and no issues from turrets.	The player becomes invisible so that the player can bypass the level with no issues from turrets.

Features	Expected Input/In Game Test	Expected Output	Actual Output
Reverse Engineering	The player is expected to either look at the Aeroblaster.py code, and find all the #TODO comments, or to read the bytecode found in the zipped dist folder if they'd like a more challenging experience	The player is able to read the code and the generated .pyc files to find the exploits in the game.	The player is able to read the code and the generated .pyc files to find the exploits in the game.
Directory Traversal	The player is expected to type into the level select input a level from a different directory so as to show they can redirect the output to be a different map than the one it's supposed to be.	The player would be able to find the cheat "x" map, and force the game to open that file from the level select input in the main menu.	The player finds the cheat command and puts it into the input level text field, allowing the user to access the x level.
Break Access Control	The player is expected to do some research on how to Break Access into a zip folder by brute forcing the password. This is considered break access due to how the player isn't supposed to be able to get into the folder in the first place	The player is able to open the zip folder and is able to view its contents with ease.	The player is able to open the zip folder and is able to view its contents.

Features	Expected Input/In Game Test	Expected Output	Actual Output
Main Menu	The player is expected to interact with this screen so as to enact the different vulnerabilities or to start the game normally.	Player is able to understand the buttons and input fields with ease.	Players can input their names and wanted levels, as well as starting the game.
Controls Screen	The player is expected to interact with this screen so as to find out the controls of the game.	Players can understand the controls of the game and proceed normally through the game.	Players can function properly in the game and maneuver the map
Pause Screen	The player is expected to interact with this screen so they can either restart the level, go back to the main menu, quit the game, or resume the level.	Players can press the letter “p” on the keyboard which will pause the game and allow the user to resume the level, restart, go back to the main menu or view the controls.	Pressing the “p” on the keyboard will allow the player to pause the game and either restart, resume, or change the level.
Name input	The player is expected to interact with the bar to input their name or to exploit it for either the SQL injection or the cryptography cipher.	User is able to input his/her name in the input field in the main menu, and the name is later stored into the score database.	Users can input their names into the proper text field and it gets stored into the database.
Level input	The player is expected to interact with the bar to input the level they want to jump to or use it to exploit it for the directory traversal.	The player is able to select which level they would like to play, and thus is able to pick which vulnerability they’d like to do.	Player can select the level they’d like to do based on which vulnerability they'd like to complete

Features	Expected Input/In Game Test	Expected Output	Actual Output
Start and End Buttons	The player is expected to interact with the buttons to either start the game to use it to quit and exit it.	The player is able to begin the game with the level they input into the level select text field, and then whenever the player would like to end the game they click on the End the game to close it.	The player is able to either start or end the game either by clicking the start button, which begins level 1, or end the game by using escape.