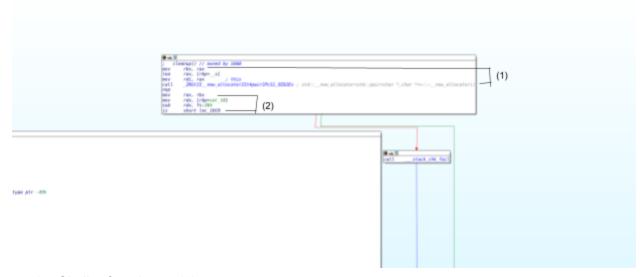


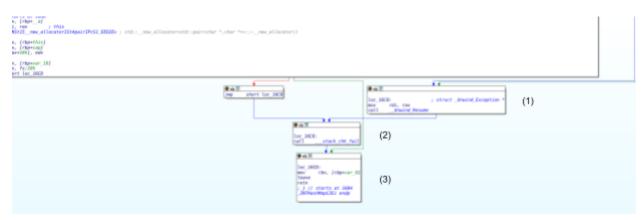
Note: NOP instructions are likely used for stack alignment.

- 1. Function Prologue, rbx is pushed onto the stack because it will be used to refer to the hashmap in later functions (i.e its own) when it is used.
- 2. Stores the memory address for the constructed map and takes in the cap argument (i.e stores it) for initializing the size of the hashmap later.
- 3. Doing research on the fs register and knowing that i used default g++ options to compile the hashmap has me believe that this is loading the stack cookie to prevent basic buffer overflows. This is further reinforced by the constant stack fail checks present within the program. Outside of basic buffer overflow safeguards, this could potentially be useful within the hashmap context as the keys themselves have no real security checks and are just char pointers which the user can input into.
- 4. Places the memory location of the hashmap into rbx and the pointer for the allocated vector that's used to store keys and values.
- 5. Sets the nullptr values for the key and value tuple and calls pair.
- 6. Attempts the hashmap initialization with the capacity and the vector initialization
- 7. Heap allocation for the hashmap class
- 8. Stores the hashmap starting memory address into rax and the capacity value in its offset (like in private declaration in the source code)
- 9. Checks for stack corruption or tampering

Cleanup Function, owned by (6) in the try, except block Likely a separate thread meant to cleanup after main was finished, since it links to the stack unwinding functions within C++.



- 1. Similar function to (8)
- 2. Similar function to (9)



- 1. Unwind continuation
- 2. Checks if canary is messed up
- 3. Function epilogue