Final Report: Buffer Overflow Vulnerability in "The Game"

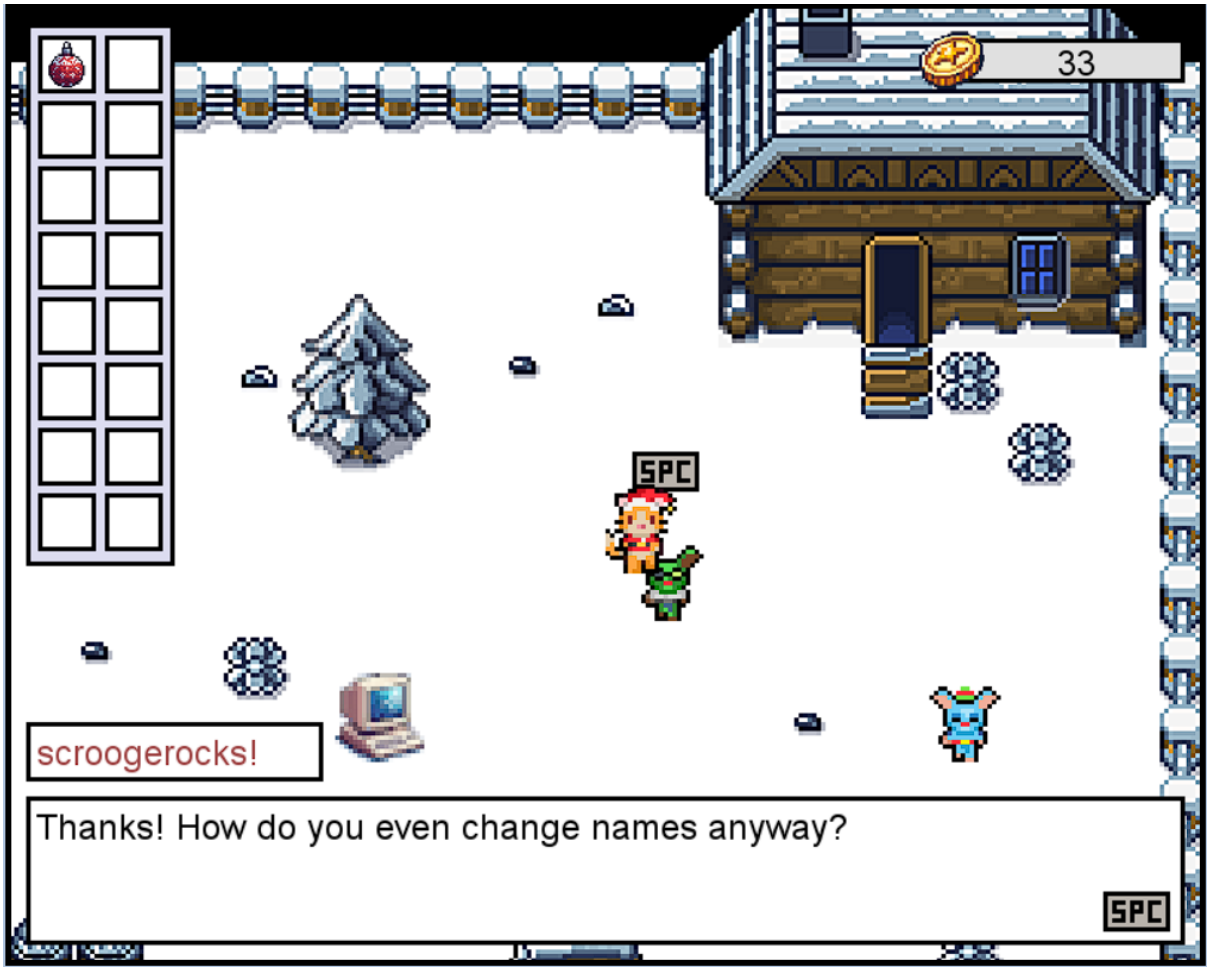
Executive Summary:

During the internship, an investigation into reported anomalies in the game "The Game" revealed a critical buffer overflow vulnerability. This flaw allows players to manipulate memory contents, leading to unintended consequences such as gaining arbitrary amounts of in-game currency and corrupting other variables. The primary cause of this vulnerability is the insufficient validation of user-inputted strings, specifically the player_name variable.

---

1. Vulnerability Discovery:

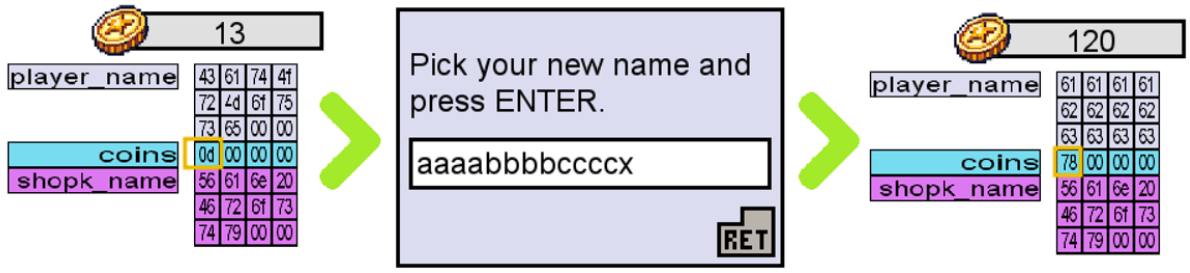
Van Jolly initially reported abnormal behavior after changing the in-game name to "scroogerocks!" and observing a sudden increase in coins. Investigation using the built-in debug panel revealed that the game lacked proper checks on the length of user-provided strings, leading to memory corruption.



The new name scroogerocks! Overwrites the memory space of **coins** , as you can see the first memory slot contains the exclamation from the name.

2. Exploitation Scenario:

The vulnerability is exploited by changing the in-game name to a string longer than the allocated memory for the player_name variable (e.g., "aaaabbbbccccx"). This results in overflow, with excess characters overwriting adjacent memory spaces, including the coins variable. As a consequence, players can manipulate their coin count arbitrarily.



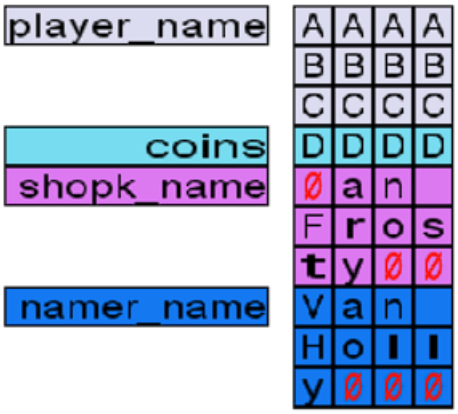
3. Technical Analysis:

The root cause of the buffer overflow is the absence of boundary checks on the player_name variable. When the string length exceeds the allocated space (e.g., 12 bytes), excess characters spill into adjacent memory, affecting variables like coins. The lack of proper validation allows unintended manipulation of in-game data.

4. Further Investigations:

Additional experiments demonstrated that the vulnerability extends to other variables, such as shopk_name. Strings exceeding the allocated space disrupt adjacent memory, leading to unexpected behavior, including empty names and altered visible player names.

Rename to AAAABBBBCCCCDDDD (16 characters).



Notice how the game adds a NULL character after your 16 bytes, which overwrites the shopk_name variable. If you talk to the shopkeeper, you should see his name is empty.

This happens because the game reads from the start of the variable up to the first NULL byte, which appears in the first byte in our example. Therefore, this is equivalent to having an empty string.

5. C++ Memory Handling:

The vulnerability arises due to the inherent nature of C++ memory handling, where boundaries of variables are not strictly enforced. The game fails to check the size of the user-inputted strings, leading to unintended memory corruption. Strings are stored in memory with a NULL character indicating the end, and overflowing strings affect subsequent variables.

6. Recommendations:

- Implement proper boundary checks for user-inputted strings to prevent buffer overflows.
- Validate and sanitize user input to ensure data integrity and security.
- Conduct thorough code reviews to identify and address potential vulnerabilities.
- Educate developers on secure coding practices, especially concerning memory handling in C++.

7. Conclusion:

The identified buffer overflow vulnerability in "The Game" poses a significant risk to data integrity and player experience. Addressing this flaw requires immediate attention to prevent exploitation. By implementing secure coding practices and enforcing proper input validation, future releases of the game can mitigate such vulnerabilities and provide a more secure gaming experience.

Reference: *tryhackme.com*