The Effect of Soft Errors on Control Loops: A Feasibility Study on Gaming Control Systems

Sabina Smajlaj

1. Control System

A control system is a set of devices that regulates the behavior of another set of devices. There are two common types of control systems: an open loop control system and a closed loop control system. In an open loop system, the output of the system is based solely on the inputs to the system. In a closed loop control system, the current output is analyzed and changes are made based on the feedback mechanism in the system.

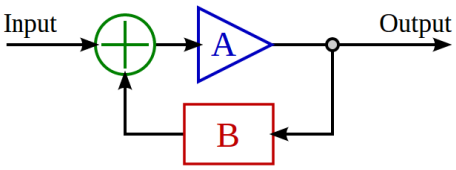

Figure 1: A simple feedback control loop

Since most control loops include a feedback mechanism, closed loop control systems will be the focus of this report. These systems have a central control loop, which typically includes sensors and the control loop algorithm, that tries to regulate a variable at a reference value. Based on the feedback results, the control loop will, for example, either increase or decrease the variable.

One common type of a closed loop algorithm is the Bang-bang control system. Bang-bang is a feedback controller that switches abruptly between two states depending on the output and the value of the feedback that it receives. They are most often used to control a system that receives a binary input, making them ideal to simulate.

Closed loop controllers are widely used today and are constructed with programmable logic controllers or microcontrollers. Since they are so widely used, their maintained security is extremely vital.

2. Security

There are various types of security breaches that these control loops could suffer from. The most basic kind is due to the general wearout of the system. As chips age, certain components such pins or transistors could permanently break or become stuck at certain values. Depending on the role these components play in the overall control loop, they could substantially alter the functionality of the control loop and compromise the system. These kind of errors are more rare due to how sophisticated processors are. Most servers have components that are built for worst-case scenarios and have built-in sensors that can detect the damaged components and quickly create a solution.

Another type of security break could come in the form of targeted software attacks. These direct attacks can alter the bytecode of the control loop and change its functionality. What specific part of the bytecode gets altered depends on what specific workload is chosen for the control loop. Possible target areas include the instructions, the data registers, the data memory, branches, and arithmetic. Since each of these different areas will potentially have a different impact on the system, the focus of this report will be to determine what that impact is and to compare the stability of the system under each separate alteration.

3. Methods

To verify the effects that specific errors could have on control loops, a simple Bang-bang control loop algorithm was created with the C programming language. C was chosen due to the fact that it could be compiled to bytecode and edited via GDB. GDB is a GNU debugger that can alter the binary of an executable file in order to change one bit and re-run that executable. Since we want to observe the overall effect of these errors on the system, we want to be able to generate a large amount of tests. In order to do so, a python script was written to generate random GDB macros based on probability values specified in a config file.

4. Results

Below is a representation of the control system and the values in its initial state before any changes were made via the GDB macros:

Table 1: The Initial Condition of the Bang-bang Control System

Initial Equilibrium	Initial Upper Bound	Initial Lower Bound	Initial Starting Temp	Avg Number of Trails to Equil (70)
70	72	68	65	4

Running the GDB macros with the Bang-bang controller showed what would result for each type of error that was being tested.

Table 2: Effect of Types of Errors on Control System

Error	Effect of Error
Instruction	System segfault
Arithmetic	System segfault
Data Registers	No observable changes yet; need to further test
Data Memory	Variable values changed significantly

Branch	Program functionality changed significantly

When the program instructions were changed (the binary value for each instruction was altered by one bit to change its value), most of the results were system segmentation faults. This could be due to the fact that the new instruction might not have been compatible with data values supplied to it. A segmentation fault is detrimental to the system and hence errors of this type could cause large security issues.

When arithmetic instructions were changed, this also resulted mostly in segmentation faults. However, when we changed arithmetic values instead, this simply changed the functionality of the system. This in itself is harmful because if the system needs to make sure the minimum temperature never drops below 40 degrees, but the value that is getting subtracted suddenly changes from 10 to 100 and causes the minimum temperature to change to -15 degrees, the system could suffer severe damage.

This is similar to the effect that changing the data memory values has. With the Bang-bang controller, changing data memory values caused the newly created temperature to become 65601 degrees, which would trick the system into thinking that that is the actual temperature it is observing and hence dramatically decrease the actual temperature. In addition, it also caused the fan state to become 4196650, which is a value the system would not know how to decipher since the fan state is expected to be a binary value.

Changing the branching had a significant impact on the functionality of the system. By changing “jne” to “je” and other similar binary alterations, the system displayed such behavior as decreasing the temperature to well below its lower bound and concluding that it had reached the optimal temperature at 68 rather than 70 degrees.

From the above analysis, it is clear that changing the data values can have the most detrimental impact on the functionality of the system. IN addition, if we change the instructions, that could potentially cause a segmentation fault and completely halt the program.

Since changing the data memory values has such a large impact, further analysis was done into what type of variables were most sensitive to alterations. The four most important variable in the Bang-bang controller are the upper-bound, which details the maximum value the system is allowed to reach, the lower_bound, which details the minimum value the system is allowed to reach, the furnace_state, which details whether the furnace (the mechanism which allows the system temperature to become higher) is on or off, and the fan_state, which details whether the fan (the mechanism which allows the system temperature to become lower) is on or off.

Table 3: The Impact of Various Variables on the Functionality of the Control System

Starting Temp	Variable Changed	New Variable Value	Avg Number of Trails to Equil (70)	Highest Temp	Lowest Temp
65	Upper bound	75	10	73	Starting temp
65	Upper bound	78	12	73	Starting temp
65	Lower bound	63	Never reaches equil	Starting temp	Starting temp
65	Lower bound	65	Never reaches equil	Starting temp	Starting temp
63	Lower bound	65	Never reaches equil	74	Starting temp
65	Lower bound	66	4	73	Starting temp
65	Lower bound	69	6	73	Starting temp
65	furnace_state	0	Never reaches equil	Starting temp	Starting temp
65	furnace_state	1	4 (only reached once)	250564	Starting temp
65	fan_state	0	4 (reached only once)	Goes up to 73 and remains there	Starting temp
65	fan_state	1	31.6	78	21

From observing the above, we can conclude that the furnace_ state and fan_state variables are very sensitive across all other variables. This holds true especially for the furnace_state variable which when set to zero never changes from the starting temperature and when set to 1 increases without bounds. The latter scenario can be especially hazardous to systems because the system needs to maintain the temperature within a specific range in order to

avoid any hardware malfunctions and this alteration would allow that temperature to keep increasing without bounds, ultimately damaging all of the hardware.

5. Conclusion

This feasibility study showed that changing single bits in the binary code of a control system could significantly both impact the functionality of the system and damage the system components. However, there still remains more testing to be done. One thing this study failed to discover was the impact changing data registers could have on the system. More testing needs to be performed in order to determine that. In addition, the python script used in this study was not completely automatic. Further work needs to be done on the script in order to be able to determine the effect each GDB macro had without actually running the macros individually.