SSVT-SmartDoor

Group 20; Elisabeth Kletsko, Nefeli Tavoulari, Denis Thiessen

September 2023

1 Bug Report

Some bugs we weren't too sure about. Those are marked in gray with respective reasoning.

1.1 Besto

1. When the door is locked, and we try to close that door now (again), it doesn't output an invalid command. Closing a door when it is locked should be an invalid state since it isn't defined in the specification. When locking a door, it needs to be closed already. And closing a closed door shouldn't be possible.
2. When the door is closed, we shouldn't be able to close it again. This is an unspecified action and should be outputting an "invalid_command" action. This happens since there is no check present if the door is already closed while closing.
3. It shouldn't be able to output a "locked" response right at the beginning. That would presume we locked it beforehand with a passcode? That isn't the case.
4. If a door is open, it shouldn't be able to be locked. While locking, there is probably no check if a door is open or closed.
5. Passcodes higher than 9999 are accepted for locks. (During Closed, Unlocked state)
6. Since there seems to be no range check in place, then passcodes under 0 would probably work as well.
7. Some commands aren't responding within the global timeout period of 0.5s. Internal global timeout is set higher than the defined 0.5 seconds.

1.2 Logica

1. When the door is opened and we try to open it again, it should output invalid_command, instead of opened.

2. When the door is closed and we try to close it again, it should output invalid_command, instead of closed. Both of these are happening since it isn't checked whether the door is already opened or closed while applying the same operation again.

1.3 OnTarget

1. When the door is locked and we try to lock it again, it should output invalid_command, instead of locked. Locking a locked door shouldn't be possible.

2. Since there are no checks in place which prevent "double locking", unlocking a door which is already unlocked could be problematic as well.

3. Passcodes higher than 9999 are accepted for locks. (During Closed, Unlocked state)

4. Since there seems to be no range check in place, then passcodes under 0 would probably work as well.

5. It tries to lock the door with no passcode as a "parameter".

6. It tries to unlock the door with no passcode as a "parameter". (It happens after it has already been locked.) Both of those behaviours are a result of no null checks inside the locking and unlocking mechanism.

1.4 quickerr

1. When the door is opened, and we close it, it should be closed instead of opened.

2. When we simulate that when ?close $\rightarrow$!opened. If we do this cycle twice, it expects an invalid_command instead of an opened. Considering the behaviour of both of those bugs, the model probably goes into the closed state, but it sends out an "opened" label. This explains the second bug since closing a closed door results in an invalid command under normal conditions.

3. It tries to lock the door with no passcode (void) as a "parameter".

4. It tries to unlock the door with no passcode as a "parameter" (void). (It happens after it has already been locked.) Both of those behaviours are a result of no null checks inside the locking and unlocking mechanism.

5. If the door is locked and we try to lock it again, then it should be an invalid command and not locked again. During the locking process, no check is in place if the door is already locked or not.

1.5 SmartSoft

1. It tries to lock the door with no passcode as a "parameter". (It happens after it has already been locked.)

2. It tries to unlock the door with no passcode as a "parameter". (It happens after it has already been unlocked and closed.) Both of those behaviours are a result of no null checks inside the locking and unlocking mechanism.

3. Passcodes higher than 9999 are accepted for locks. (During Closed, Unlocked state, when locking should be allowed) Since there seems to be no range check in place, then passcodes under 0 would probably work as well.

1.6 TrustedTechnologies

1. When the door is locked with a passcode and we try to unlock it with a different passcode, it should output incorrect_passcode, instead of unlocked. There is no check in place if the provided unlocking passcode is actually the correct one.

1.7 univerSolutions

1. It tries to lock the door with no passcode as a "parameter". (It happens after it has already been locked.)

2. It tries to unlock the door with no passcode as a "parameter". (It happens after it has already been unlocked.) Both of those behaviours are a result of no null checks inside the locking and unlocking mechanism.

3. Passcodes higher than 9999 are accepted for locks. (During Closed, Unlocked state)

4. Since there seems to be no range check in place, then passcodes under 0 would probably work as well.

5. The passcode "0" acts as an invalid passcode. No correct range-check has been implemented, zero is not included in the passcode range.

$$(0 < x <= 9999)$$

   instead of

$$((0 <= x <= 9999))$$

6. Unlocking with a different passcode provides an "!invalid_passcode" instead of an "incorrect_passcode"

1.8 XtraSafe

1. Passcodes higher than 9999 are accepted for locks. (During Closed, Unlocked state)
2. Since there seems to be no range check in place, then passcodes under 0 would probably work as well.
3. It tries to unlock the door with no passcode as a "parameter" (void). (It happens after it has already been locked.)
4. Since there seems to be probably no null-pointer check in place, locking the door with no passcode could also be possible.

2 Model file

Link to SmartDoor model file. (https://gist.github.com/D45Hub/06f61a158a87ce17f556fd1686ced8ca)