On the Turing Completeness of the Wind Test Bed

Brianne Dunn

The past week we worked on the Modbus control system used in ODU's Wind Test Bed project to manipulate its environment to create simple unintended programs. My example was a timer that would periodically in increments of ten seconds send a codeword message to attackers with the idea that any codeword could signify a particularly vulnerable time. This would only occur after a flag was set that marked the machine as being vulnerable (in this example it was simply at boot.) In addition, my colleagues created programs that would send messages with read receipts, and average calculators. The Modbus environment that we were given in our library gave us flexibility to complete programs that were translatable to simple python.

Turing completeness is typically attributed to a system that, with unlimited resources, would be able to compute any problem that a computer would be able to. These characteristics typically comprise of memory storage, reading and writing capabilities, and recursive conditional logic. Given the span of all our projects, we were able to use each of these features of the Modbus environment.

For storage we used the Modbus registries and coils, retrieving data from them later in functions like **repeat_until()** which allowed us to continuously loop through statements while also checking Boolean conditions such as the data assigned in **machine.setflag(...) (eg. "is vulnerable", True.)** This ability to work through sets of data and make different decisions based on the outcome of that data, makes this system that we're working with Turing complete. Outside of its intended function, we can put together programs that, given infinite resources, would be able to become more complex and like a computer in function.

This is relevant to our project as this exploitation of the Modbus system illustrates pushing it to become a Weird Machine that can behave in wildly unintended ways. Though the system may not practically have infinite resources, it's Turing completeness shows that it wields the capability to act more complex and efficiently in direct correlation with its resources. In our example we constructed simple programs using the Test Bed's own logic, but on a larger scale this could grow to be more damaging on critical infrastructure systems that have a greater abundance on memory and power.