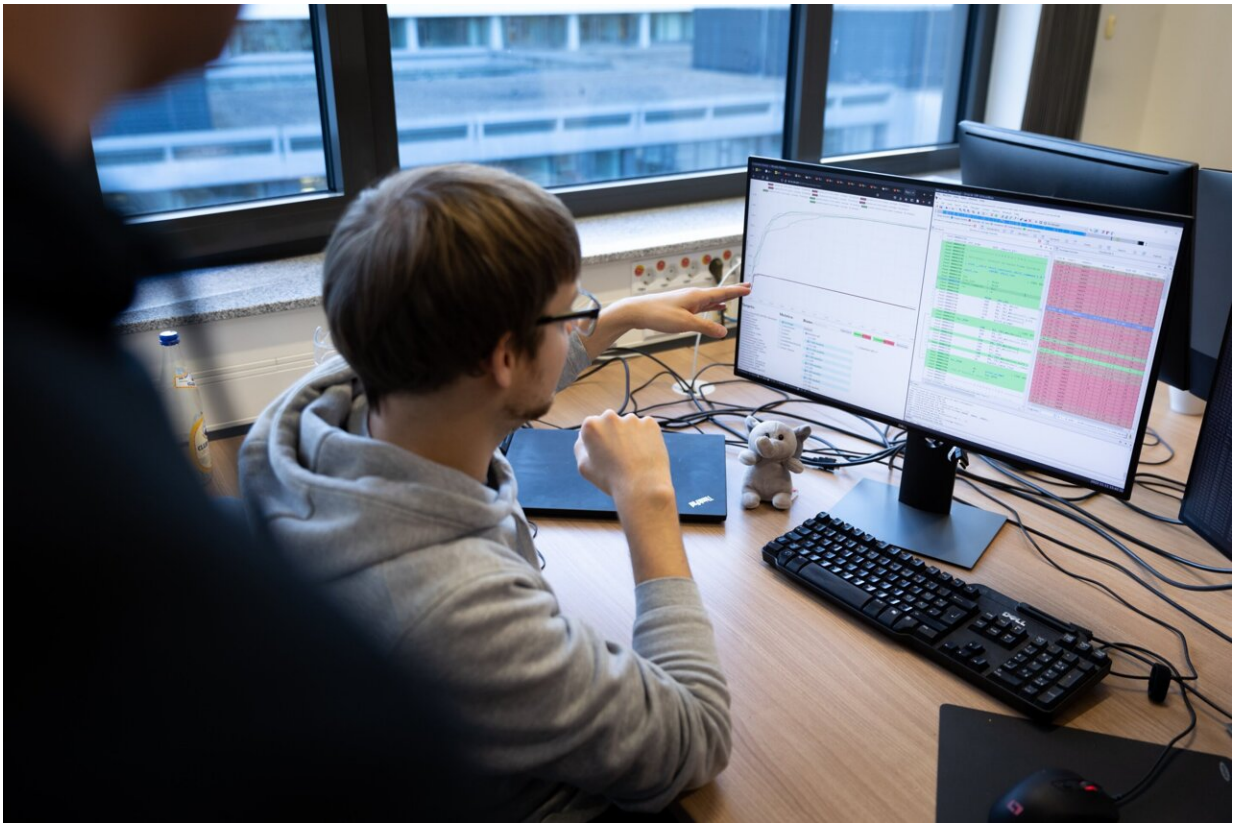
Identifying software vulnerabilities quickly and efficiently

December 14 2022



The researchers evaluate their fuzzer's code coverage, i.e. how much of the program code can be analysed with their tool. The result: The code coverage is by a factor of 4 higher than with other algorithms. Credit: Michael Schwettmann

Almost every new program code has bugs that, in the worst case, can

potentially compromise security. In order to detect them quickly and efficiently, researchers from the Horst Görtz Institute for IT Security at Ruhr University Bochum, Germany, have developed a new system called Fuzzware.

It specializes in analyzing embedded systems, i.e., mini-computers that can be found in smart light bulbs, intelligent thermostats and industrial control systems, to name but a few. *Rubin*, the Ruhr University's science magazine, published an article on their work.

The Bochum Ph.D. student Tobias Scharnowski, supervised by Professor Thorsten Holz, has presented the results at the 31st Usenix Security Symposium in the U.S. in August 2022. He conducted the research in cooperation with colleagues from the University of California Santa Barbara and the Vrije Universiteit Amsterdam.

Crashing the software on purpose

The group uses what is known as fuzzing to detect errors in program code. Fuzzers are algorithms that feed the tested software with random inputs and check whether they can crash the application with them. Such crashes indicate programming errors. The fuzzer keeps varying the input in order to explore as many program components as possible step by step.

Fuzzing is already established for certain areas of application, for example to test operating systems such as Windows or Linux. It has not yet been widely used to test embedded systems, however, because they pose a number of challenges: the software—the so-called firmware—is embedded in a piece of hardware with which it interacts. Often the systems have relatively little memory and slow processors. This is a problem if the researchers want to carry out fuzzing directly on the system. It would take far too long to try out all possible inputs and wait

for the system's response.

Virtual imitation of hardware

This is why the team doesn't analyze the firmware directly in the industrial control unit or in the light bulb. Instead, they recreate the hardware virtually—this process is called emulation. The emulator makes the firmware believe that it is inside the real device. For this, it has to interact with the program in exactly the same way as the real hardware would.

In order to accelerate the procedure, the researchers add another step to the fuzzing process by narrowing down the possible inputs. First, they model the framework in which the inputs must be located in order to be logical for the firmware. For example: if the hardware is a refrigerator with a temperature sensor, the refrigerator hardware can report the measured temperatures to the refrigerator's software, i.e., its firmware. Realistically, it's not possible for any given temperature to occur, it has to fall within a certain range. Therefore, the firmware is only programmed for a certain temperature range. It could not process other values at all, so there is no need to fuzz them.

Limited inputs facilitate efficient analysis

Together with colleagues from Santa Barbara and Amsterdam, the Bochum team tested 77 firmwares using Fuzzware. Compared to conventional fuzzing methods, they sorted out up to 95.5% of all possible inputs.

This enables Fuzzware to check up to three times more of the program code than conventional methods in the same amount of time. In the process, the group also identified additional vulnerabilities that had

remained undetected with other fuzzing methods.

More information: Fuzzware: Using Precise MMIO Modeling for Effective Firmware Fuzzing. www.usenix.org/conference/usen … entation/scharnowski

Provided by Ruhr-Universitaet-Bochum