RUHR-UNIVERSITÄT BOCHUM

RUB

RUBIN

SCIENCEMAGAZINE

SPECIAL ISSUE

IT SECURITY

Three tough nuts for quantum computers to crack

This is how artificially generated images reveal their true colours

Start-up: Ready for the new generation of mobile communications

#33 Volume

Special Issue 2023

Fuzzing

INTELLIGENT **MONKEYS**

Researchers from Bochum are particularly quick at finding security vulnerabilities in IT systems. Their trick: they focus on the essentials – and explain it with the theorem of the infinitely typing monkeys.

A program code is a bit like a jungle: complex in structure, difficult to view from the outside, with countless paths that can be taken through it. Finding vulnerabilities in such code is like looking for animals among the trees in the jungle: you know they are there, but you can't see them directly. This is why PhD student Tobias Scharnowski is developing new methods to efficiently detect programming errors in the jungle of ones and zeros. He is conducting research at the Chair of System Security at the Horst Görtz Institute of Ruhr University Bochum, supervised by Professor Thorsten Holz.

The researchers are primarily interested in embedded systems: "We are trying to increase the security of computers that most people don't even know are computers at all," explains Scharnowski. Examples of such embedded systems include smart light bulbs, refrigerators connected to the internet and intelligent thermostats, to name but a few. All these objects contain electronic control technology with many lines of program code in which errors may have crept in. But household appliances are not the only things on the IT experts' agenda. Above all, they are interested in industrial control systems, for example in critical infrastructures such as energy supply. These are areas where security gaps could have dramatic consequences.

Scharnowski and Holz use 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 challeng-

i EMBEDDED SYSTEMS

An embedded system is a combination of hardware and software that serves a specific purpose within a larger system – in a car, for example, this includes electronic controls of the seats. An embedded system is essentially a computer that serves a narrowly defined purpose.

SOFTWARE, HARDWARE, FIRMWARE

Hardware is the term used to describe all devices in the computing sector; unlike software and firmware, it exists in the physical world. Software and firmware, on the other hand, are programs that only exist in the virtual world. Firmware is a specific type of software that is used to control hardware, i.e., it fulfils a precisely defined purpose for a given piece of hardware.

FUZZING

Fuzzing is a method used for identifying vulnerabilities in software. In the process, the software is fed many different inputs and run until an input causes it to crash. A program crash indicates a bug.

The IT specialists search for errors in the programming code of firmware, a specific type of software that is needed for the control of hardware.

es: the software – the so-called firmware – is embedded in a hardware with which it interacts. Researchers usually have little information about the hardware and how it works. "It's like a black box for us," describes Thorsten Holz. In addition, this black box is usually not particularly powerful – 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. This is why the team doesn't analyse 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. "This means we have to imitate all the interfaces that exist between hardware and firmware," explains Thorsten Holz. Once this is accomplished, the researchers can test the firmware in a powerful system. Still, it would take a long time if they let their fuzzer try out all theoretically conceivable inputs. That's why 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: let's assume that 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 just 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.

"We only use the inputs in the fuzzing process that the firmware expects and can handle," points out Thorsten Holz and compares the process to the Infinite Monkey Theorem: "This theorem states that, if you let monkeys type on a key- ▸

> IF A SYSTEM HAS NEVER BEEN TESTED WITH FUZZING, IT WILL HAVE UNDISCOVERED VULNERABILITIES.
>
> Thorsten Holz

board for long enough, they would eventually come up with the works of Shakespeare." The same applies to the fuzzer: if you let it try again and again, it would, by chance, eventually use meaningful inputs. But it would take a long time. "We want to make our monkeys a bit more intelligent, though," says Tobias Scharnowski. "We take away all the keys they don't need and try to get them to press only useful keys. With the inputs that are left, we can still test the code all the way down." This makes fuzzing with the Bochum system – known as Fuzzware – particularly efficient.

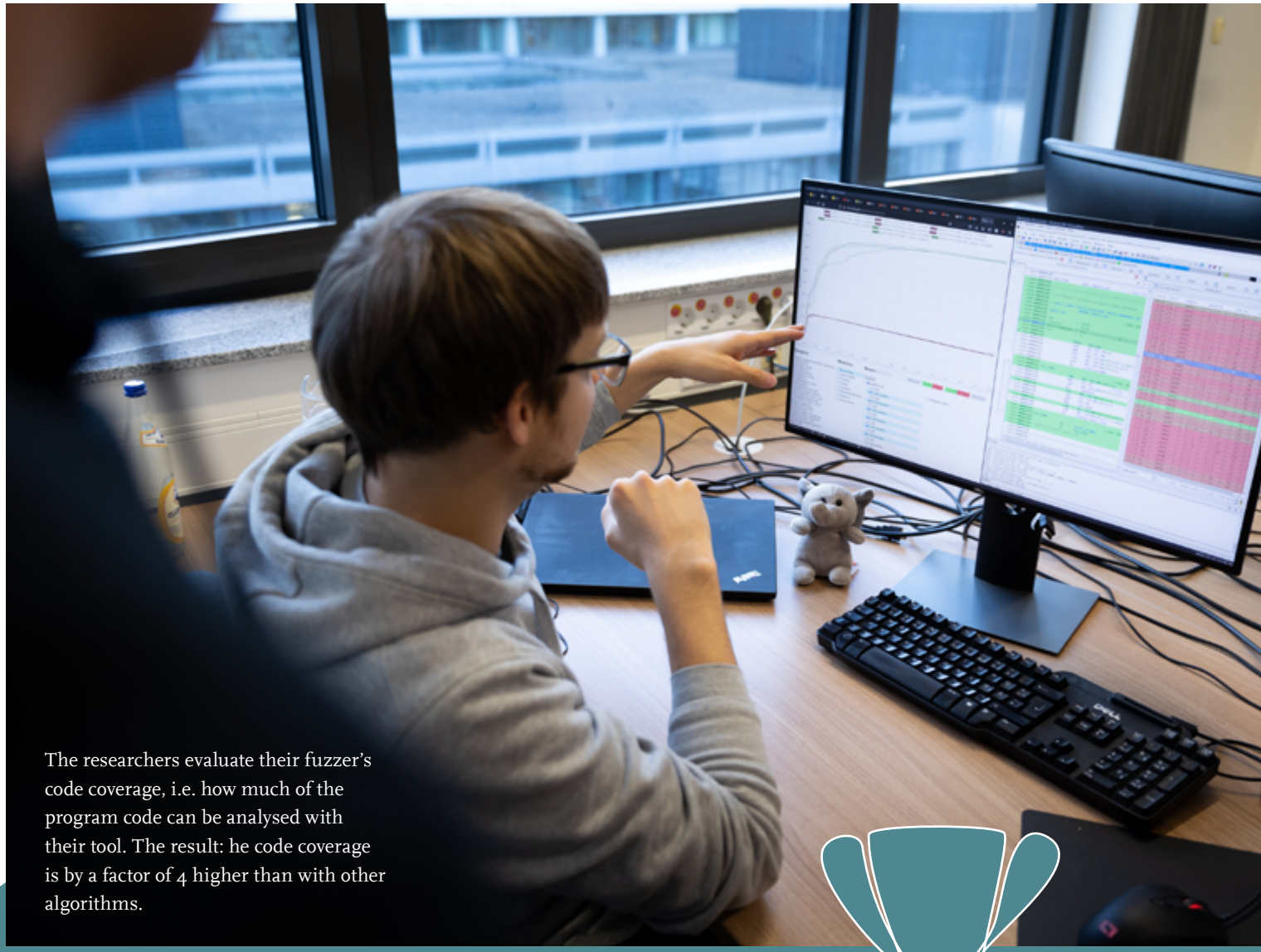
Together with colleagues from Santa Barbara and Amsterdam, the Bochum team tested 77 firmwares using Fuzzware. Compared to conventional fuzzing methods, they sorted out up to 95.5 per cent of all possible inputs. This enables Fuzzware to check up to three times more of the program code than conventional methods in the same amount of time. In the process, the group also identified additional vulnerabilities that had remained undetected with other fuzzing methods. "You can always find something," says Thorsten Holz. "If a system has never been tested with fuzzing, it will have undiscovered vulnerabilities."

In the case of embedded systems in particular, it is almost impossible for programmers to create the perfect code. "In order to talk to the hardware of embedded systems, you have to use a low-level programming language," explains Tobias Scharnowski. For many applications, programmers can't simply fall back on code snippets that have been developed for other applications. They have to build their code from scratch. Edge cases – namely states that the system rarely encounters – may then not be taken into account. "For our fuzzers, however, these states are easy to analyse," says Scharnowski. "They can therefore help make the systems more robust." By reporting any vulnerability they identify to the manufacturers, the researchers contribute to greater security in industry, light bulbs and refrigerators, to name but a few.

text: jwe, photos: ms

Tobias Scharnowski is PhD student at the Horst Görtz Institute for IT Security.

For many years, Thorsten Holz was one of the Principal Investigators of the Cluster of Excellence CASA.



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

Interview

"A RIPPLE WENT THROUGH THE CROWD"

Software companies are delighted when researchers find bugs in their code before attackers do. They even organise bug-finding competitions. The Bochum team has already won plenty of prizes.

Mr. Scharnowski, people in your field often talk of bug bounties. What do you mean by that?
It's a kind of bonus programme offered by software companies for detecting vulnerabilities. The more serious the vulnerability you discover, the higher the prize. Some manufacturers even run competitions.

Have you ever taken part in one?
In 2020, I entered the Pwn2Own competition in Miami together with some colleagues. It was organised by various manufacturers from the industrial security sector and was about devices that control industrial plants. One of the elements we attacked was the so-called DNP3 protocol, which is used for communication between control systems, for example in the critical energy sector. We were the only ones who managed to reach the highest category for this task and gained complete control over the program.

That sounds like a remarkable success.
Yes, that was quite an exceptional experience. The competition had different targets, and it started with an announcement of which team would tackle which target. When our idea was presented, a ripple went through the crowd.

And what did you win?
Between the three of us, we received 87,500 US dollars in prize money. It gives us the freedom to buy software and equipment for our next adventures of this kind.

jwe

EDITOR'S DEADLINE

The rabbits in the CASA Universe are startled: the seemingly well-secured access to Rabbit Mark's carrot stash has been hacked and all winter supplies have been stolen. The brave bunny Betty then starts looking for support at the nearby CASA Hub C – a mysterious place that is supposed to hold solutions for digital security. Thus begins the adventure of Betty the bunny, the protagonist of the first science comic from the Cluster of Excellence CASA. Along with Betty, the readers explore the Research Hub and learn about the research priorities and challenges that the scientists in the Research Hub C "Secure Systems" deal with on a daily basis. Find out how to read this and more CASA comics at no cost at:

↗ *casa.rub.de/en/outreach/science-comics*

Answers
DEEP FAKE-QUIZ
The following faces are real:
1a, 2a, 3b, 4a, 5b, 6a

LEGAL NOTICE

PUBLISHER: Cluster of Excellence CASA at the Horst Görtz Institute for IT Security at Ruhr University Bochum in collaboration with the Corporate Communications Department at Ruhr University Bochum (Hubert Hundt, v.i.S.d.P.)

EDITORIAL ADDRESS: Corporate Communications Department, Editorial Office Rubin, Ruhr-Universität Bochum, 44780 Bochum, Germany, phone: +49 234 32 25228, rubin@rub.de, news.rub.de/rubin

EDITORIAL BOARD: Dr. Julia Weiler (jwe, editor-in-chief); Meike Drießen (md); Lisa Bischoff (lb)

PHOTOGRAPHER: Michael Schwettmann (ms), Dammstr. 6, 44892 Bochum, Tel.: +49 177 3443543, info@michaelschwettmann.de, www.michaelschwettmann.de

COVER: Sashkin – stock.adobe.com

PHOTOGRAPHS FOR TABLE OF CONTENTS: Michael Schwettmann

GRAPHIC DESIGN, ILLUSTRATION, LAYOUT:
Agentur für Markenkommunikation, Ruhr-Universität Bochum,
www.einrichtungen.rub.de/de/agentur-fuer-markenkommunikation

PRINTED BY: LD Medienhaus GmbH & Co. KG, Van-Delden-Str. 6-8, 48683 Ahaus, Germany, Tel.: +49 231 90592000, info@ld-medienhaus.de, www.ld-medienhaus.de

EDITION: 800

DISTRIBUTION: Rubin is published twice a year in German language; the regular issues are available from the Corporate Communications Department at Ruhr-Universität Bochum. The magazine can be subscribed to free of charge at news.rub.de/rubin/abo. The subscription can be cancelled by email to rubin@rub.de. The special issue 2021 is available from the Horst Görtz Institute for IT Security. In case of interest, please contact hgi-presse@rub.de.

ISSN: 0942-6639