WHAT'S ALL THE FUZZ ABOUT?!

What's all the fuzz about?

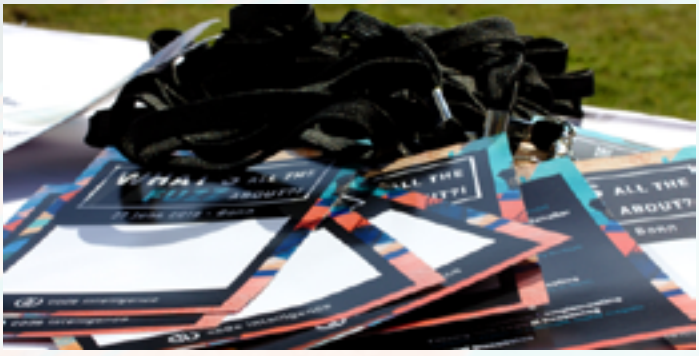
This IT security event was organized and hosted by Code Intelligence in Bonn on June 27, 2019.

The event's objective was to discuss current projects and latest developments in cybersecurity with a focus on fuzzing. Leading experts in the field were invited on this day to give talks and lead discussions on fuzzying and similar technologies. There was also plenty of room for networking and one-on-one discussions.

The presentations took place in the Digital Hub co-working area located in Bonner Bogen, an up-coming area of Bonn overlooking the Rhine river. Code Intelligence additionally made its company rooms located in the same building available for the breakout sessions.

The event brought together IT security specialists, lead developers, CTOs and IT project managers from different industries. Four talks on a wide range of topics regarding IT security were delivered as well as three breakout sessions conducted in small groups.

Various drinks, fruit, and snacks were provided as refreshments. A local Indian food truck served lunch and the event finished off in a relaxed atmosphere with a barbecue and some beers.



Keynotes

After registration and a brief round of introduction, participants were offered two keynote speeches. One of them was derived from personal experience and the other one was based on academic research papers.

How to Manage Crises and Flourish

The event kicked off with a speech by Marc Wallert about managing crises in digital environment. Marc Wallert is a keynote speaker and resilience coach with 17 years of experience working in global companies. Marc shared his personal experience of being kidnapped by Islamic terrorists at the age of 27. He was captured together with a group of tourists in Malaysia, brought to the south of the Philippines and held hostage for 140 days, surviving harsh dschungel conditions and the constant threat of being killed by terrorists.

Foto: Marc Wallter

Marc explained how he and other hostages were able to survive through supporting each other and acting as a team. He went on to talk about how companies and individuals can use the same strategies for crisis management.

When facing a crisis, the first important thing is to accept the situation at hand: it makes no sense to block out the problem or to go through alternative scenarios. You should also try to think in a positive and optimistic way. However, being too optimistic may cause the company or the individual to become euphoric thus preventing them from taking the necessary preventive or corrective steps. Another important strategy is strengthening the self-efficacy through helping others or taking any kind of action, no matter how small.

The result of correct crisis management should be not just to overcome the crisis but also to learn from it and use it as a chance for development.

Academic View on Fuzzing

The next speech, by Prof. Thorsten Holz, was titled "How I Learned to Stop Worrying and Love Fuzzing". Thorsten Holz is a professor at Ruhr University Bochum. His research interests are in the field of applied aspects of secure IT systems and automatically detected software vulnerabilities. Furthermore, he is a speaker of the Cluster of Excellence ‚CASA - Cyber Security in the Age of Large-Scale Attackers".

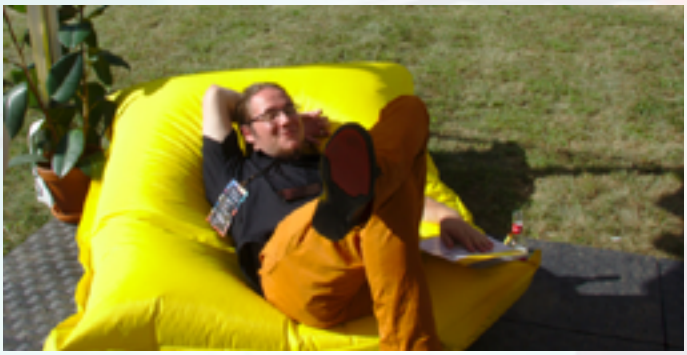
Foto: Prof. Dr. Thorsten Holz

Thorsten started his talk with a brief introduction of his research team and the topics they focus on: static and dynamic code analysis and especially fuzzing. Next, he described several papers on fuzzing that his team has published in the past years.

- Redqueen is an engine based on fuzzing with input-to-state correspondence. Since the majority of existing fuzzers rely on coverage feedback and thus have drawbacks (e.g requiring the source code and not recognizing nested hashes), there came an idea to create a fuzzing engine that does not only feed inputs into a program but also influences the current state of the program through these inputs. Inputs that did not trigger any interesting behavior in the program are discarded, and the fuzzer thus "learns" the required input format. Redqueen is based on the same architecture kAFL tool uses. As an example, it enables fuzzing OS kernel components with a high level of efficiency.
- Nautilus engine concentrates on searching for deep bugs in software using grammar structure input. It recognizes how the program tree is structured and at the same time uses a feedback loop to generate inputs to achieve a better code coverage, i.e. reach as many edges and blocks of the program tree as possible.
- Grimoire is a current project that attempts to bridge the gap between Redqueen and Nautilus (for cases when there is no grammar available, i.e. blackbox fuzzing). It tries to generalize and synthesize structure of inputs while fuzzing.
- AntiFuzz is being developed to make program code resistant to fuzzing. This can be used, for example, by companies to protect their code when making it available to customers.

Thorsten summarized his talk with mentioning some open questions and challenges regarding fuzzing in academic world.


Four Europeans freed



Breakout Sessions

After a short break, event attendees were offered three breakout sessions running in parallel. These sessions were repeated in the afternoon, so that each attendee got a chance to participate in two sessions out of three.

Fuzzing 101

The first breakout session explained the basics of fuzzing technology and included some practical examples. The speaker, Dr. Khaled Yakdan, is a co-founder of Code Intelligence and an expert in binary code analysis. He also worked as a malware analyst at Fraunhofer FKIE in addition to his PhD. He deals with fuzzing on a binary level.

In the first part of his talk, Khaled provided an overview of different software testing methods (static and dynamic analysis as well as symbolic execution) and presented the advantages and disadvantages of these methods. In the second part, he focused on fuzzing as a dynamic software testing technique. Fuzzing monitors the performance of the target software while providing random, invalid or unexpected inputs to it. After briefly talking about the history and development of the fuzzing method, Khaled went on to talk about two main fuzzing methods: "dumb" fuzzing conducted with random inputs and coverage-based fuzzing that includes information about code structure and a seed corpus of inputs aiming to reach different parts of the code structure.

Foto: Dr. Khaled Yakdan

Next, the speaker described the fuzzing process using the example of libFuzzer - from writing fuzz targets to checking final results. The workshop included a live demonstration of testing several programs using fuzzying methods. He also explained how sanitizers (tools for error detection) work and how fuzzing can be improved through using a seed corpus of inputs and token syntax dictionaries. The workshop finished off with an overview of current challenges in fuzzing and how they can be solved.

Fuzzing for Embedded Systems

The second break-out session, "Challenges of Fuzzing for Embedded Systems" was conducted by Sirko Höer from Code Intelligence GmbH. Sirko is responsible for vulnerability research in embedded systems and has over 10 years of experience in the area of cybersecurity.

He started his presentation with an explanation of what embedded systems are. He also brought some hardware to show how these systems physically look. After a brief excursion into the main fuzzing principles and types of fuzzing techniques, he stated that the main problems of fuzzing on IoT devices are:

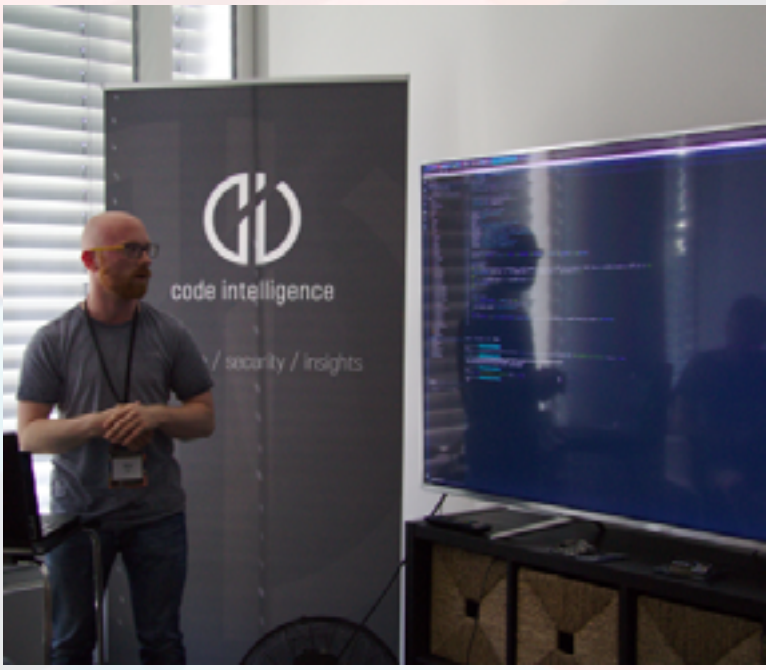


Foto: Sirko Höer

- The need to cross-compile the code
- There are hardware limitations (e.g. slow processors & limited memory space of fuzzed devices)
- Many IoT devices have their own operating system

At this point, Sirko described the approach for embedded systems that is currently being developed by Code Intelligence GmbH. The essence of this approach is breaking the testing into modules, creating corresponding fuzz targets and fuzzing them separately. During this process, a seed corpus of testing inputs can be derived that can be used for integration tests and later on for system tests.

Sirko went on to introduce QEMU emulation software that can be applied in testing embedded devices. This was followed by a live demonstration of modifying libFuzzer for arm processor testing and fuzzing with QEMU User mode. The result was that QEMU user mode had better performance than running the tests on SoC devices but had less executions per second for most of the fuzzed modules. After running module tests as unit tests, integration tests are also conducted in QEMU environment and system/hardware tests are normally run by the manufacturer of the device.

The presentation was followed by a lively discussion on introduced approach and what limitations and advantages it can have in practice.

Fuzzing in Practice

The third workshop was run by Sergej Dechand, one of the founders of Code Intelligence. He is an expert in usability aspects of IT security and worked as a project manager at Fraunhofer FKIE Institute.

Sergej dedicated his workshop to explaining the goal of Code Intelligence as a company. He started with an overview of software development process and specifically an overview of the testing process within it. He mentioned different testing methods used by the companies: static analysis, unit testing and penetration testing.

Fuzzing, together with symbolic execution, belongs to the domain of penetration testing. Proper penetration testing is challenging in practice because a lot of developers neglect testing due to the lack of time and the lack of expertise regarding the available security tools. On the other hand, external penetration testers lack the domain knowledge and need to gain insight into the software under test, which causes high expenditures and slows down the project.

Sergej stated that the tool offered by Code Intelligence (CI Security Suite) offers easy access to modern testing techniques. It also contains a plugin that integrates into the IDE of the developer for easy set-up of fuzz tests. The speaker provided some examples on how this tool can be used in practice, for example for fuzzing of server applications. The session closed with an extensive question round regarding the functionality and features of the CI Security Suite.

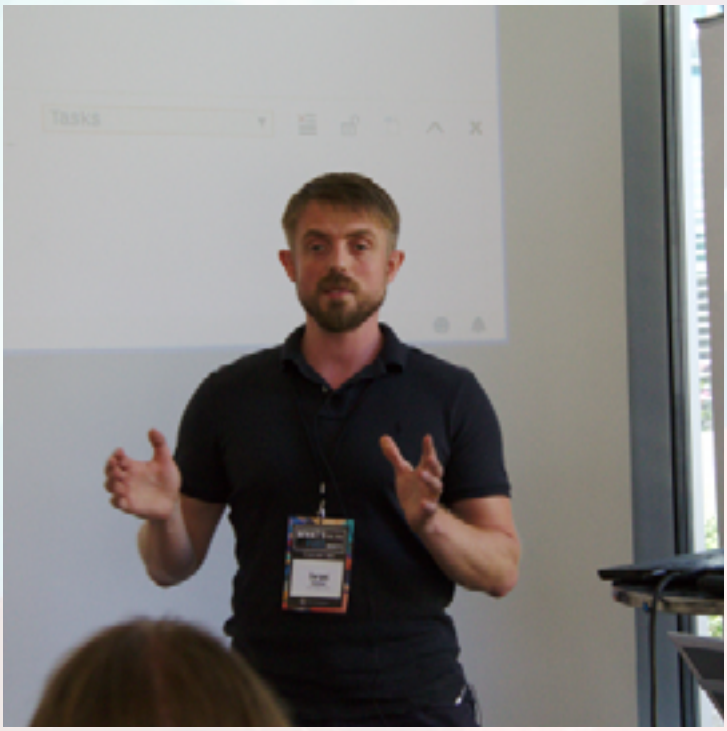


Foto: Sergej Dechand

Keynotes

n the afternoon, two more keynote speeches were offered. The first one examined the human component in software design and testing, whereas the other one focused on research in the field of symbolic execution testing.

Human-Centric Software Testing

After lunch, the event continued with a presentation on human-centric software testing by Prof. Matthew Smith. Matthew is an internationally known expert for usable security and privacy. He is a professor at Bonn University and a member of Fraunhofer FKIE Institute in Bonn. Recently he has also become a member of the ‚Wise Council of Cybersecurity".

In the first part of his presentation, Matthew gave a brief introduction on usable security and privacy. Usable security examines how developers can make their software and interfaces more usable for the end user without compromising on security.

Foto: Matthew Smith

Mathew highlighted that a lot of developers tend to focus on their main product goal and neglect security. Two cases were provided as examples: HTTPS certificate usage on Android devices and a study conducted with students and professional developers on implementing user password protection in software development process.

In the second part, the presenter covered a qualitative study on security warnings. A focus group of 33 participants were interviewed on what kind of warnings they preferred when writing code, how often and when these warnings should be displayed.

The final part was an outlook on how artificial intelligence and usable security can be combined to drastically improve software testing in the future.

Symbolic Execution: Complementing Fuzzing

The last presentation of the day was conducted by Sebastian Pöplau. Sebastian Pöplau is a PhD student in the Software and Systems Security group of Eurecom (France), under the supervision of Aurélien Francillon. In his PhD research, he focuses on the security of embedded devices, moving from low-level software to hardware aspects. His presentation was called "Symbolic Execution: Complementing Fuzzing with Logical Reasoning".

Foto: Sebastian Pöplau

After introducing himself, he gave a definition of symbolic execution. As opposed to fuzzers, which generate inputs traditionally without taking code structure into account, symbolic execution tools precisely capture the computation of each value. They use solvers at each brunch to generate new inputs and thus to provide the precisely calculated input to cover all parts of code.

Sebastian moved on to describe symbolic execution in more detail using the example of KLEE. It is an open source instrument that runs on bitcode, usually compiled from source code by clang compiler.

KLEE explores the program and generates test cases to reproduce any crashes it finds.

Though symbolic execution in theory can find inputs for any feasible path, it is still rather slow compared to fuzzing and requires complete source code and a lot of work to set up. There have been attempts to combine fuzzing and symbolic execution, for example in a tool called Driller.

Sebastian also provided an overview of other engines for symbolic execution, mostly further developments of KLEE, which can be used for various practical purposes. According to the speaker, symbolic execution remains an area of active research.

Conclusion

"What's all the fuzz about?" was the first big event on fuzzing topics organized by Code Intelligence and it turned out to be a success. The event was conducted in an open atmosphere and offered a lot of possibilities for discussion and networking.

Here is some feedback provided by the participants:

> "The talks were of high quality and demonstrated the depth of technical knowledge."
>
> "I found the presentations to be very interesting. They motivated me to learn more about the topic of fuzzing."
>
> "I liked the personal and casual way of communication with the organizers before and during the event."
>
> "I liked everything - from the way how the agenda was conveniently printed on the back of the badges to the selection of topics for presentations and workshops."
>
> "The event provided insights on how to easily integrate fuzzing into software development process, showcased new applications for fuzzing and state-of-the-art technologies".
>
> "I enjoyed the event and the barbecue afterwards."
>
> "A big thanks to the orga-team for preparing the event and ensuring its smooth running."

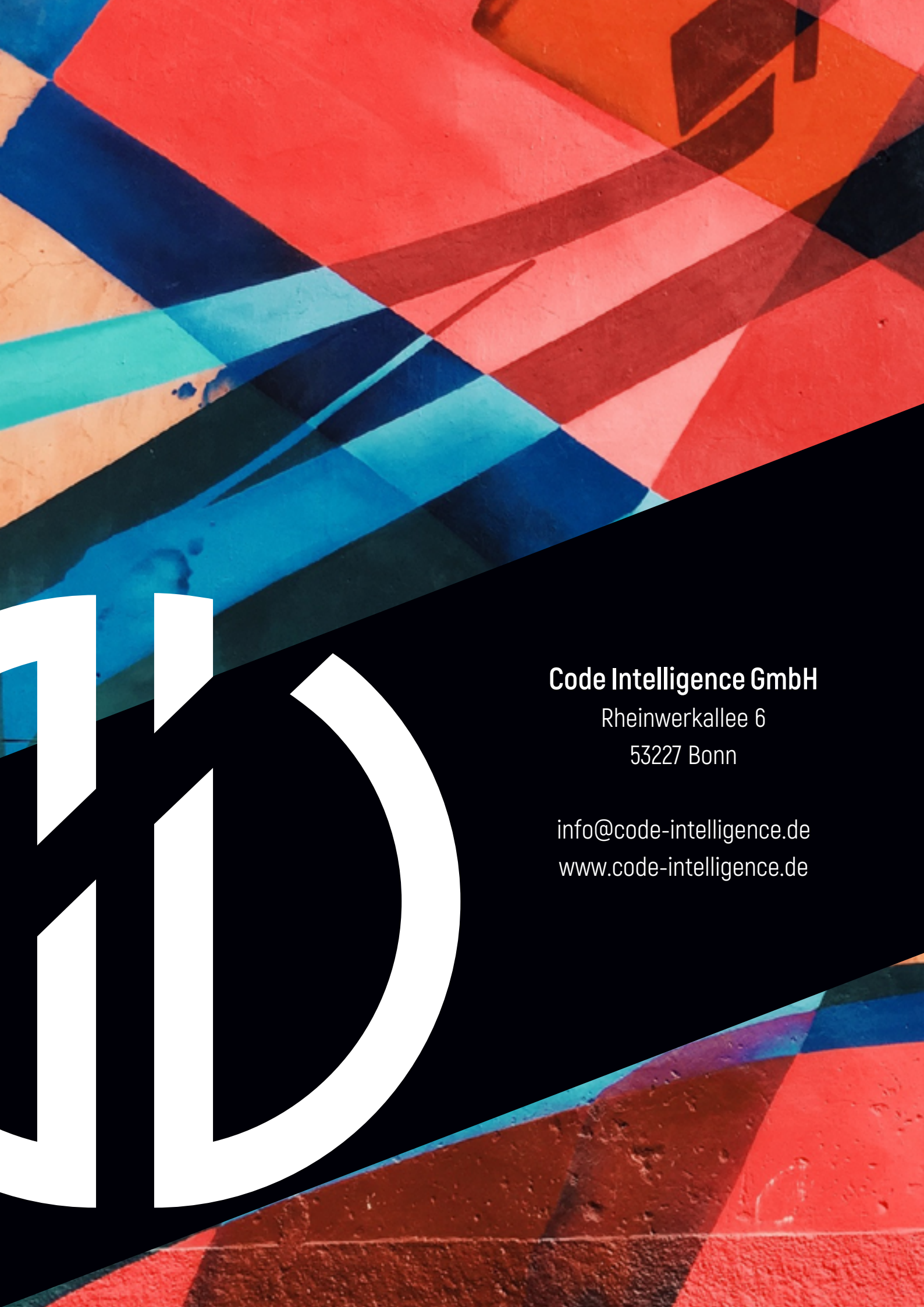
Get in touch

Jonathan Reimer

+49 228 2869 5830

sales@code-intelligence.com

www.code-intelligence.com



Code Intelligence GmbH
Rheinwerkallee 6
53227 Bonn

info@code-intelligence.de
www.code-intelligence.de