code intelligence

Simplify your Application Testing

STUGRM#10:
“Fuzzing für Tester: Effizient und einfach sichere Software entwickeln”

Brief introduction

Alexander Weise
Vice President

Sales specialist and strategic thinker with experience in the security industry

Code Intelligence
GmbH

Vision - Easier access to modern software testing techniques for everyone

www.code-intelligence.com

Digitalization comes with growing pains

3 weeks of heise.de news

Mitsubishi-Hack: Sicherheitslücke in Anti-Viren-Software als Einfallstor

Es gibt neue Details über die Hacker-Attacke auf Mitsubishi Electric. Mittlerweile ist die Sicherheitslücke bekannt und was die Angreifer kopiert haben.

27.01.2020 UPDATE

Uni Gießen nähert sich nach Hacker-Attacke wieder dem Normalbetrieb

Aufgrund eines IT-Sicherheitsvorfalls war die Universität Gießen um Weihnachten 2019 zeitweise komplett offline. Nun gehen erste Dienste wieder online.

06.01.2020

Jeff Bezos gehackt: Forensische Analyse weniger ausführlich als möglich

Die Forensiker, die das mutmaßlich gehackte Smartphone von Jeff Bezos untersucht haben, haben wohl nicht alle Mittel ausgeschöpft. Ihr Report ist öffentlich.

23.01.2020

MDhex: Angreifer könnten medizinische Geräte von GE Healthcare kontrollieren

Aufgrund von unsicheren Standardeinstellungen und veralteter Software mit Sicherheitslücken ist die Überwachung von Patienten gefährdet.

24.01.2020

DKB-Kunden können nicht auf ihre [illegible] zugreifen

Der Server-Dienstleister der Online-Bank DKB wurde angegriffen. Deshalb ist dessen Online-Banking beeinträchtigt.

08.01.2020 UPDATE

#heiseshow: Riesiges Datenleck bei Buchbinder – Was hat das für Folgen?

Die Autovermietung hat wochenlang Terabyte an Kundendaten frei zugänglich im Netz stehen lassen. Was es damit auf sich hat, besprechen wir in der #heiseshow.

23.01.2020

IT-Systeme abgeschaltet: Automobilzulieferer Gedia kämpft mit Trojaner

Der deutsche Autoteile-Hersteller ringt mit den Folgen einer heftigen Hacker-Attacke. Offenbar hat ein Erpressungstrojaner viele Daten verschlüsselt.

23.01.2020

Jetzt handeln! Exploit-Code für kritische Citrix-Lücke gesichtet

Es könnten Angriffe auf Citrix Application Delivery Controller und Gateway bevorstehen. Bislang gibt es nur einen Workaround. Patches sollen folgen.

13.01.2020 UPDATE

What do all these stories have in common?

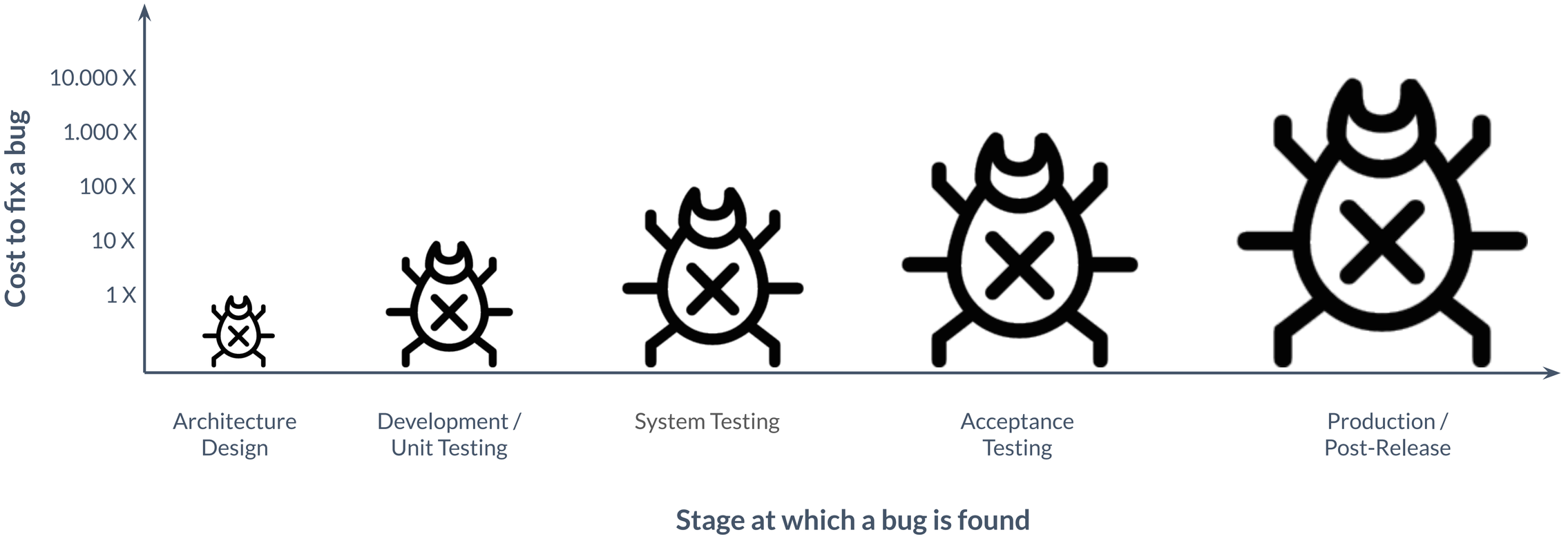
Software Bugs

leading to critical vulnerabilities

costing 1.7 Trillion USD[1] for Top 606 Software Defects

[1] https://www.tricentis.com/software-fail-watch/

Why are bugs that expensive?

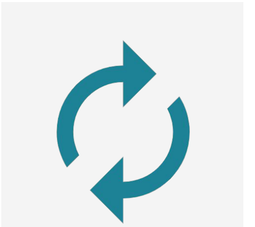


Market trends triggering different coping strategies

Security Testing Market Trends

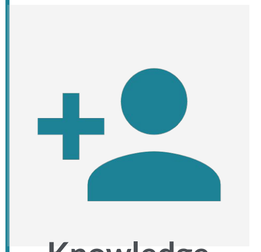
- Internet of Things
- Web and mobile applications / Customer data
- AI based solutions
- Advanced persistent threats evolving
- Adoption of open source security applications
- Deployment of third-party applications
- Government mandates and regulatory compliances

Automation & Cultural Changes



Processes

from serial to integrated (CI)



Knowledge

from human to AI-assisted

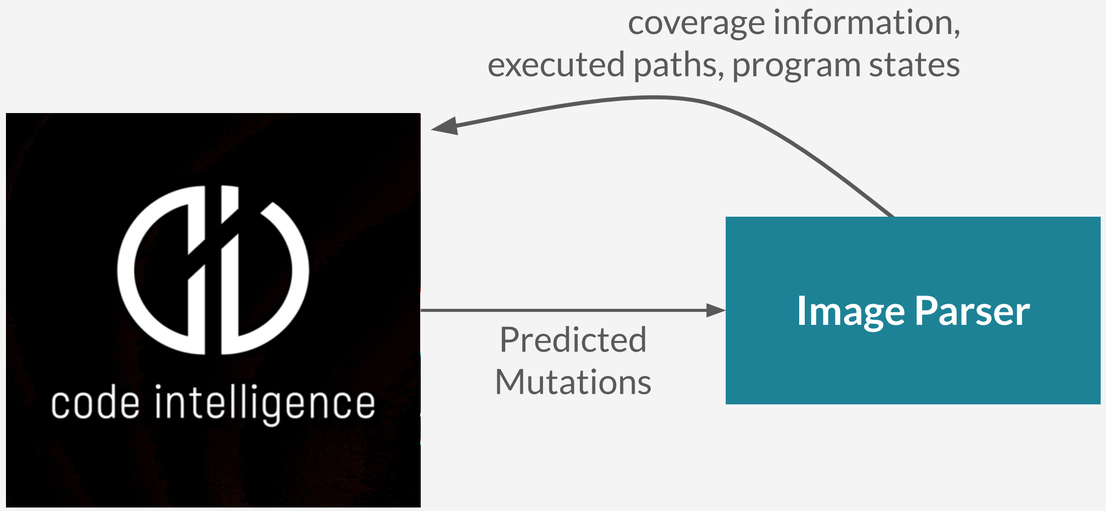
Toolbox

from academic to everyone

Core of Our Technology: AI-Assisted Fuzzing with Feedback Loop

Modern Fuzzing uses instrumentation for feedback to smartly mutate inputs to detect bugs and vulnerabilities

Example: Fuzzing an image parser application



***Click image** for animation*

History of Modern Fuzzing

2015 AFL introduced feedback-based fuzzing in academia: not only using randomize inputs but adding smart algorithms generating inputs increasing code coverage significantly

2019 With modern fuzzing, over 16,000 bugs have been discovered in Google Chrome and 11,000 bugs in 160 open-source projects[1]

2020 Until today fuzzing could only be used by experts. Therefore, we developed *CI Fuzz* that is both technologically advanced (combination of AFL, libFuzzer, honggfuzz and our custom solution) while easy-to-use for developers

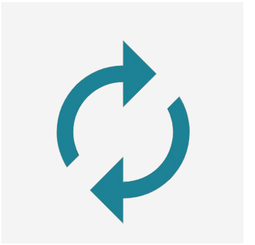
More about modern fuzzing here.

Market trends triggering different coping strategies

Security Testing Market Trends

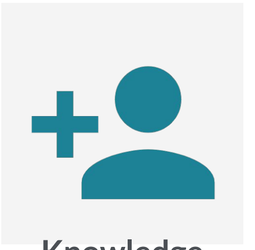
- Internet of Things
- Web and mobile applications / Customer data
- AI based solutions
- Advanced persistent threats evolving
- Adoption of open source security applications
- Deployment of third-party applications
- Government mandates and regulatory compliances

Automation & Cultural Changes



Processes

from serial to integrated (CI)



Knowledge

from human to AI-assisted

Toolbox

from academic to everyone

Application security testing toolbox: Different approaches

1. Manual Testing

- Manual revision and exami[illegible]ode
- Used by security experts w[illegible] special security tools

human effort

2. Static Application Security Testing (SAST)

- Analysis of application source code, byt[illegible] coding and design conditions
- Code is not exec[illegible]

automated pattern recognition

3. Dynamic & Interactive Application Security Testing (D&IAST)

- Test during code execution detect[illegible] false positives to nearly zero
- Analysis of outpu[illegible] & continuous use as input for subsequent test runs

automated code execution

4. Feedback-based Application Security Testing (FAST)

- Software under test is fed with input[illegible] ring testing
- Fuzze[illegible] during execution

automated code execution + AI-assisted feedback

Challenges in Early Software Security Testing Wasting Time and Nerves

Unit / Integration / System Testing

enormous overhead in manual effort covering mostly functional testing

slow and expensive due to enormous manual effort

not covering security testing properly

Static Code Analysis (SAST)

enormous amount of false positives and high amount of overseen bugs

usability issues due to false positives

various bugs uncovered too late in the process due to false positives

Dynamic Testing (DAST / IAST)

either requires too much manual work or doesn't achieve proper code coverage

still require enormous manual effort to integrate properly

very limited code coverage leading to missing bugs

hard **to understand**

There is Hope: Smart Fuzzing Superior in Bug Discovery

Vulnerabilities and bugs found automatically even in professional environments

- Google (August 2019): **>14,000 bugs found in 200** oss projects
- Google (October 2019) : >**16,000 bugs in Google Chrome** with fuzzing

BUT: Practical Limitations

- Hard to integrate in existing development environments
- Tooling from fuzzing experts for experts

1: https://github.com/google/oss-fuzz
2: https://opensource.googleblog.com/2019/02/open-sourcing-clusterfuzz.html

CI Fuzz Testing Platform Enables Effortless Software Security + Stability

Effortless / Scalable

- No initial inputs required → auto deduction of types
- No changes to existing build and development processes
- Easy interface definition
- Almost no false positives due to actual execution
- Never leave your IDE (setup, stacktraces, load erroneous inputs into debugger + setting breakpoints)

Smart SDLC Orchestration (Fast Deployments)

- Agnostic CI/CD integration
- Intuitive reporting
- Code coverage visualization
- Most common web backend languages
  - JVM (Java, Kotlin…), Go
- Most common embedded device / IoT languages
  - C/C++, Rust

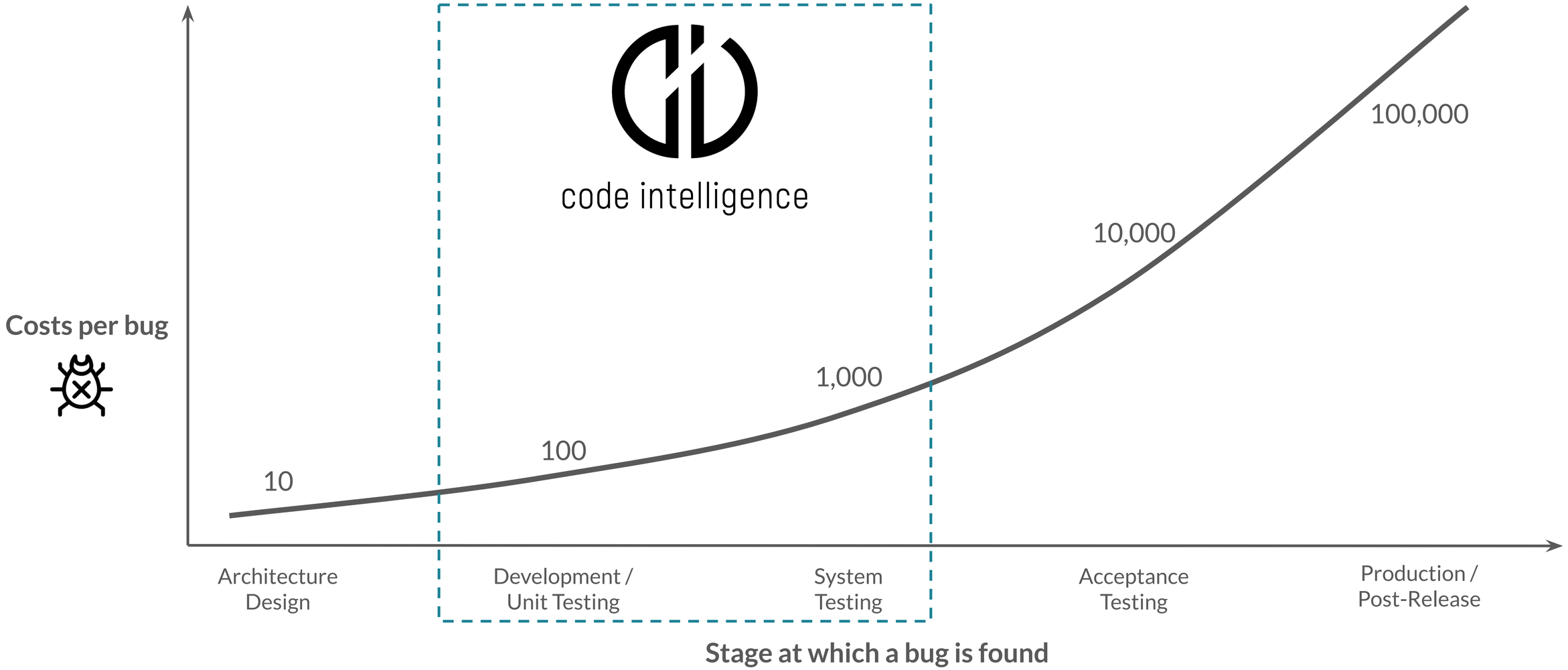
Highest Code Coverage (Validation)

- White-box fuzzing allows coverage maximization based on source code
- Structure-aware input generation covering large parts of the business logic
- Protocol-based fuzzing with auto detection (stateful)
- Smart combination of OSS fuzzing engines, e.g., AFL or libFuzzer

Smart Bug Detection

- Sophisticated bug and vulnerability detection aiming for low false positives
- Smart combination of various bug detection frameworks e.g., Google Sanitizers, ZAP
- Most common CWEs supported (incl. OWASP Top 10 and API Top 10)

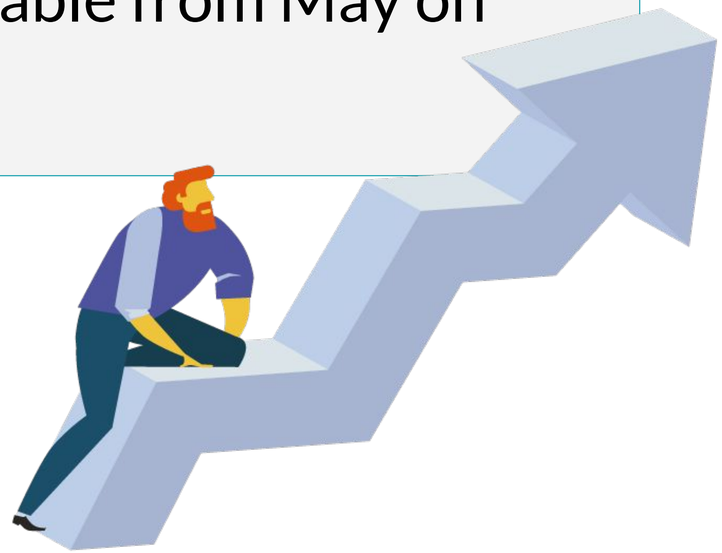
Finding Bugs Early Avoids Follow-up Costs / Reduces Time to Market



* The “rule of ten” is well-founded by the results of several studies in Japan, the USA and Great Britain, which dealt with the causes of product and quality defects.

Testing of Web Backends in Logistics

- **12 highly critical vulnerabilities** and **>50 bugs** leading to unknown errors
- Drastically reduced implementation costs and testing effort
- Information leakage checks not performed but available from May on



Testing of an Intrusion Detection System

- **12 critical vulnerabilities** (CVEs)
- Already using OSS Fuzz (libfuzzer + AFL)
- Some bugs found by our network protocol fuzzer

Sponsored By

Thank you for your attendance!

code intelligence

Code Intelligence GmbH
Rheinwerkallee 6
53227 Bonn

Alexander Weise
+49 170 / 5473 061
weise@code-intelligence.de
www.code-intelligence.de