

Is there something hidden in the code?



Gavin Wilsher describes an example of where undesirable code appears to have been deliberately inserted into a train control system rendering it inoperative after maintenance at a 3rd-party site. He explores whether the safety community should do more to ensure these types of issue do not find their way into safety-related / critical software whether deliberate, inadvertent or through the increasing use of Artificial Intelligence.

This article was prompted by a well-publicised incident that occurred on a particular train type in Europe in 2023 [1]. Readers of this newsletter may already be familiar with the story. There are certain similarities between this story and the well-publicised issue involving vehicles with diesel engines having hidden code to ensure that emissions tests could be passed.

It was reported on the RailTech website in January 2024 that the train manufacturer was taking legal action against the maintenance company [2] although it is not clear what progress has been made. All information in this article directly related to the incident is in the public domain.

The intention here is not to analyse that particular incident but rather to look under the surface and wonder if there is a more endemic issue in the railway sector (and potentially other safety-critical sectors) that should prompt more safety interest. Hopefully, this will serve to get a conversation going (even if it is simply to tell me that I might be worrying about nothing).

This article presents a summary of the incident I refer to above, but is only informed by it and not truly about it. It is really about things that can be hidden in “the code” and whether we should be concerned. However, I think it is important to understand what appears to have happened, how it was investigated and subsequently resolved before concluding whether there is a wider problem.

Nobody can be sure of the motivations for the train manufacturer to do what they appear to have done, although some have speculated that this was done as a direct result of them not winning the maintenance contract in a competitive tendering situation. Looking at the issue superficially, it is easy to reach this conclusion, but the fact is that nobody outside the company can be sure. It is certainly not the purpose of the article to wonder why they did it.

The following few paragraphs summarise what was reported to have happened.

The Issue

An Impuls 45WE train, manufactured by Polish company Newag, that was undergoing mandatory, scheduled maintenance by a recognised maintenance organisation unexpectedly failed to run once the maintenance had been completed. The train appeared to be fine, it just failed to run.

The Lower Silesian Railway in Poland runs eleven of these types of trains and, according to the maintenance schedule, another one was due to be sent for maintenance while the first – instead of returning to passenger service – was still sitting in an apparently failed state in the workshop.



The second train then underwent the same maintenance as the first with the same resulting outcome – it failed to run following completion of the maintenance. The maintenance team continued to try to get the first train running, without success. It is reported that the manufacturer was approached for assistance but they declined to help. If this is indeed the case, it raises questions about the management of contracts, which could have been written such that there were obligations placed on the manufacturer to support maintenance or other technical issues.

This scenario continued to play out on further trains, including one at another maintenance depot belonging to the maintenance company.

The maintenance organisation was conducting the maintenance in accordance with the manufacturer's instructions, provided in a 20,000 page manual. At the completion of the maintenance it was expected that the train would run and be returned to passenger service. However, the inverters failed to supply voltage to the motors and no one had any idea why this was happening. Maintenance technicians conducted deep investigations but failed to find the cause.

The problem came to the attention of the press because timetables were affected and replacement trains were needed in order to maintain some level of service.

The maintenance company was incurring substantial penalties by this time and was anxious to get to the bottom of the problem.

At some point they reached out to a group of hackers to see whether they might be able to help get into the train's computers to understand what was happening.

Many months of analysis and reverse engineering ensued and they eventually uncovered some rather interesting conditions written into the software code of the trains. What they discovered was the GPS co-ordinates of maintenance depots hidden in the code. Furthermore, there was a condition written into the code that prevented the train from running if it spent more than 10 days at one of these locations. They also found the GPS co-ordinates of a depot belonging to the train manufacturer, but this had a different set of conditions associated with it.

The whole thing was presented at the Chaos Communication Congress in December 2023. A video of the presentation is available in [3]. It is well worth watching.

So what?

There is no suggestion in any of the commentary about this issue that there were any safety implications arising from it. The immediate impacts were potentially to the maintenance organisation's reputation and their bottom line. It is also the case that passengers seem to have been affected by the timetable changes and overcrowding on the smaller, replacement trains that needed to be deployed. Having said that, overcrowding can have a minor safety impact.

However, the real concern is that manufacturers can hide such information within code that is presumably subject to internal code reviews, without challenge. The hidden information was not highlighted in the extensive maintenance manual, despite it being directly related to maintenance, so the documentation could be seen to be incomplete i.e. why wasn't there a warning in the maintenance manual that after 10 days in the depot the train may fail to run? The information related to the depot belonging to the train manufacturer was presumably in the code so that the conditions could be tested. If that is the case then it can be assumed that if the software was subjected to internal Verification and Validation (V&V), that it passed without anyone requesting the removal of what appears to be information that is unrelated to correct functioning of the software. Indeed, effort may have been expended testing that the conditions associated with this additional information actually functioned as intended. That is probably less concerning because at least it would indicate some rigour was applied to making sure this apparently unnecessary information performed as intended.

"The real concern is that manufacturers can hide such information within code that is presumably subject to internal code reviews, without challenge"

In the train scenario described above there is no direct relationship with safety. However, what it illustrates is just how easily seemingly unnecessary information can be hidden away in the code. What if something like this was added to the Safety Integrity Level (SIL) 4 <insert any high integrity safety-critical software from any sector here> code? Can we assume that, provided it passes internal V&V then the software will be deployed onto the market?

Under specific conditions the hidden information may lead to an error condition that causes a safety hazard. Failure to find these conditions during code analysis and test does not prove they don't exist, all it does is prove that the testing failed to find them.

Questions have been raised on this matter on various online forums along the following lines:

- How was the train manufacturer able to do this?
- Why isn't there an independent overseer auditing and certifying that what the computer code does is exactly as documented?
- Why was the train manufacturer not obliged to send source code to the railway operator?

These sort of questions do seem to point to the standards and regulations and what they mandate (or do not mandate) in terms of independent review.

Taking this issue at face value, I have become convinced of two things:

- We should be concerned that this sort of problem may end up crossing over into high integrity software intended for safety applications at some point
- This is probably not the first time something of this nature has happened, it is just that this time, it has been made very public

Set aside for a moment that the issue that prompted me to write this article was created by humans acting in a deliberate manner; there is currently growing emphasis on using so called Artificial Intelligence (AI) in software engineering as a 'co-pilot' and in writing test cases. This drives me to wonder whether this might start to become much more of a problem, especially given the current propensity of AI to 'hallucinate' i.e. make things up. The AI may insert random artefacts into the code that could, under certain conditions, cause a safety hazard but not be detected during code analysis and test, particularly if the same AI engine is used to manage the testing or analyse the code.

I am convinced that this is something we need to start to think much more carefully about. There has already been significant thought about this by the SCSC's Safe Autonomous Systems Working Group.

What could we do about it?

So, do we need a more rigorous and complete approach to independent (third-party) code review? We could argue that following standards like IEC61508 [4] and applying the techniques and measures defined therein should be sufficient. But is it? Software safety standards often mention redundant code but perhaps they also need to consider the assurance of the absence of non-required or even adversarial code.

I have experienced a situation where SIL4 software used in a train control system was developed correctly in accordance with EN50128 [5], did exactly what it was supposed to do, passed all the module and system level tests but still produced an unsafe failure condition when deployed onto the railway. Fortunately, this was detected by a vigilant signaller before any harm was done.

“following standards like IEC61508 and applying the techniques and measures defined therein should be sufficient. But is it?”

The particular condition in the software that caused this issue could very easily have been the result of the addition of geolocation data such as that described in the Impuls 45WE scenario. With the benefit of hindsight, this now adds to my belief that we need to do more than we currently are doing to protect against this.

I would like to see an independent, international working group set up, comprising representatives from commercial organisations and the regulators, to explore practical ways to deal with potentially unnecessary information being hidden in the code. Although all of the suggestions I make in this article involve different approaches to independent code review, the working group should also consider whether there are other approaches that could be used i.e. activities other than code review.

The working group could then publish informal guidance in the form of a community standard and work towards getting the guidance introduced formally into international standards to make it mandatory.

Some thoughts about the type of questions the working group could address are presented in the following paragraphs.

Should the independent assessor be responsible for a full, independent code review? That sounds expensive and would depend upon the assessor having a comprehensive understanding of not only software engineering but the multiplicity of programming languages that may be used in safety-critical applications.

Perhaps the relevant regulatory body should be responsible for conducting sample based code reviews e.g. select software modules and perform a full review of the code? These samples could be taken using a risk-based approach to ensure that targeted effort is applied where the consequences of apparently unnecessary information being hidden in the code are the greatest. The temptation might be to use the highest SIL to select the samples and this sort of approach would be most useful where there are multiple SIL 4 software modules to select samples from.



Perhaps the end user (train operator, airline, power station operator etc.) should be responsible for undertaking code reviews? But surely they depend upon the independent assessor, the regulator and even the manufacturer themselves to assure them that the system is safe enough to put into use?

It would be easy to jump to the conclusion that the answer to whether more third-party reviews should happen is 'yes' but that may be because, in the world of safety-critical systems, we tend to be risk averse and more checking tends to make us feel more comfortable.

So, what about the standards and regulations? Perhaps they should be updated to mandate fully independent review of such code if the code can be accessed by any safety-related applications (either intentionally or otherwise).

Thinking about my concerns regarding the use of AI in software engineering, the company I work for has very strict guidelines about how, when and to what extent AI can be deployed. I have no idea whether we are unique or whether that is a standard approach. However, I do think that a cross-industry standard should be mandated that sets out certain rules regarding the application of AI to software engineering in safety-critical fields. I do not believe that the International Standard on AI, ISO/IEC 22989 [6] does this for us. Perhaps an update to IEC61508 is required. Currently it states (in IEC61508-3) that AI is not recommended at SIL 2 and above. AI has moved on a lot since the current version was published in 2010.

Concluding Remarks

This article has sought to question whether, in light of the train maintenance issue described above, the safety community (us) need to be considering more rigorous third-party reviews of software code. I have also used it to raise my concerns about the growing use of AI in software engineering and whether there needs to be a standard that covers its application in safety-related / critical software.

In writing this article, I did not set out to answer these questions (it would be arrogant of me to suggest that I have all the answers) but to raise some awareness and pose some questions that might start a conversation that does indeed lead to answers.

I often provide independent safety assessment services as part of my current role and this issue has certainly impacted the way I think about assessing the software and has prompted me to ask more probing questions about code reviews and what they have considered.

References

- [1] <https://badcyber.com/dieselgate-but-for-trains-some-heavyweight-hardware-hacking>, accessed April 2024
- [2] <https://www.railtech.com/policy/2024/01/03/poland-hackers-used-to-unblock-trains>, accessed April 2024
- [3] https://media.ccc.de/v/37c3-12142-breaking_drm_in_polish_trains, accessed April 2024
- [4] International Electrotechnical Commission (IEC), "IEC61508:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems"
- [5] CENELEC "EN50128 Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems" 2011+A2:2020 (superseded by EN50716:2023)
- [6] International Organization for Standardization (ISO), "BS EN ISO/IEC 22989 Information technology – Artificial intelligence – Artificial intelligence concepts and terminology," 2023.

Image attribution:

top image: AI generated: MidJourney

Newag Impuls train at station in Węliniec: ID 171096643 © Oxygen64 | Dreamstime.com

Code review: ID 49028202 © Rasmus Ursem | Dreamstime.com

Gavin Wilsher

Gavin Wilsher is a systems and safety engineer with over 25 years' experience with safety-critical systems across many regulated industry sectors. This experience includes safety engineering and independent safety assessment in the civil nuclear, military aerospace and railway signalling sectors.

The author retains copyright of this work.