If you think the millennium bug was a hoax, here comes a history lesson

December 31 2019, by Nir Oren

Credit: Helen Stebakov

It's not hard to find echoes of the late 1990s in the zeitgeist. Now as then, impeachment is on many peoples' minds, and films such as *The*

Matrix and *The Sixth Sense* continue to influence culture. Another feature of the same era that perhaps has a more important, if subtler, influence is the infamous Y2K bug.

Y2K was the great glitch in computer systems that looked capable of destroying civilisation at the stroke of midnight on the millennium. In the end, however, nothing much went wrong. Some people started to wonder if we had been misled all along. In fact, they couldn't have been more mistaken. Y2K is in danger of becoming one of those moments in history from which exactly the wrong lessons have been drawn.

Many of the systems that were at risk from the Y2K bug dated from the 1970s, 1980s and early 1990s. This was the era when the alleged insistence by Bill Gates that "640k [of RAM] ought to be enough for anybody" was still ringing in people's ears. Even powerful servers had only a few megabytes of RAM—a fraction of what you would find in most ordinary PCs today.

With so little space, programmers were always trying to come up with ways to conserve memory. Dates were one of those things that were integral to most computer programmes, and years came to be stored as a number between "0" and "99"—so for example, "80" would represent 1980. The advantage was that only a single byte of memory would be used. But with the new millennium soon to come around, it meant that the year "99" would become "100". As a result, computer programs would believe that the year was 1900 rather than 2000, which threatened to raise serious problems.

Bug on out

It looked likely that financial transactions such as accrued interest would be calculated incorrectly. Monitoring software would suddenly believe it had expired and ceased to work, while navigation software would not be

able to compute positions correctly. Still more alarming, failures in individual mission-critical systems might cascade. This could cause power grids, telecoms networks and financial systems to fail; oil rigs to stop pumping oil; hospital patient record systems to start prescribing the wrong drugs.

The sheer scale of such failures would make recovery difficult. This would potentially affect countries' economies and the wellbeing and even lives of people around the world. As the US president, Bill Clinton, told an audience during a speech in 1998: "This is not one of the summer movies where you can close your eyes during the scary parts."

The computer industry's response involved a massive software rewrite, with official "Y2K ready" certification issued after extensive testing. Different solutions were implemented for different systems, depending on their memory capacity. The best option was to store years as four digits. Where that was not possible, programmers might instruct a system to treat, say, dates between "00" and "50" as being in the 2000s, and years between "51" and "99" as being in the 1900s. This at least allowed systems to keep functioning.

More problematic were embedded systems where the Y2K issue existed in hardware rather than software. In such cases, the only solution was to replace the hardware itself. Estimates of the total cost for Y2K preparation came in at around US$300 billion, or about US$460 billion (£351 billion) in today's money – plus a few more billion spent on addressing issues as they arose after the turn of the century.

The big easy?

When the fateful day came and went with little more than minor problems, the questions started. A view took root that Y2K had been overblown—perhaps, for instance, to guarantee a giant pay day for

programmers. People could point to the fact that some countries, such as South Korea and Russia, had got away with doing little to mitigate the problem, not to mention small businesses.

But this ignores the fact that software patches for the bug were rolled out worldwide. Those who didn't prepare were protected thanks to the efforts of those who did. There is ample evidence, thanks to preparedness exercises, code reviews and the like, that if not addressed, the impact of Y2K would have been much more significant.

Unfortunately, the contrarian view has wormed its way into other important areas of policy. Climate change deniers and anti-vaccination activists often raise the lack of impact of the Y2K bug as evidence that experts are not to be trusted. If we are eventually successful in dealing with problems like climate change in future, don't be surprised if similar arguments about wasted time and effort appear.

By that time, the same people will probably also be able to point to a couple of sequels to the millennium bug that didn't come to much either. As I mentioned above, ancient software systems still exist which treat all dates with two digits greater than "50" as occurring in the 1900s. While most of them should be retired before we get to the next danger year of 2050, the likes of mission-critical systems can be notoriously long-lived.

We can also look forward to the year 2038 problem. This relates to the fact that Unix systems historically stored dates and times as sequences of 32 ones and zeros, interpreted as the number of seconds since January 1, 1970. When 2038 rolls around, this number will overflow for the same reason the Y2K bug occurred. Such Unix systems again form the foundation of many mission critical pieces of software.

The Unix community is well aware of this bug, however, and most of these systems will again have been replaced long before 2038. So just

like with Y2K, if the world survives these future problems, it will not have been because it was all hype. The more boring truth is often that a stitch in time saves nine. Sorry to be the bearer of good news.

Provided by The Conversation

Citation: If you think the millennium bug was a hoax, here comes a history lesson (2019, December 31) retrieved 7 May 2024 from https://techxplore.com/news/2019-12-millennium-bug-hoax-history-lesson.html