Case Study: Lethal Software Defects - Patriot Missile Failure

by Michael Barr

During the Gulf War, twenty-eight U.S. soldiers were killed and almost one hundred others were wounded when a nearby Patriot missile defense system failed to properly track a Scud missile launched from Iraq. The cause of the failure was later found to be a programming error in the computer embedded in the Patriot's weapons control system.

On February 25, 1991, Iraq successfully launched a Scud missile that hit a U.S. Army barracks near Dhahran, Saudi Arabia. The 28 deaths by that one Scud constituted the single deadliest incident of the war for American soldiers. Interestingly, the "Dhahran Scud", which killed more people than all 70 or so of the earlier Scud launches, was apparently the last Scud fired in the Gulf War.

Unfortunately, the "Dhahran Scud" succeeded where the other Scuds failed because of a defect in the software embedded in the Patriot missile defense system. This same bug was latent in all other Patriots deployed in the region. However, the bug was masked by this fact--a particular Patriot weapons control computer had to continuously run for several days before the bug could be revealed and result in the hazardous condition of failing to track a Scud.

The official post-failure analysis report by the U.S. General Accounting Office (GAO IMTEC-92-26) entitled "Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia" provides a nice concise write-up of the problem. Included in this report is prefatory background on how the Patriot system is designed to work.

The hindsight explanation of this tragedy consists of the following:

> *a software problem "led to an inaccurate tracking calculation that became worse the longer the system operated" and states that "at the time of the incident, the [Patriot] had been operating continuously for over 100 hours" by which time "the inaccuracy was serious enough to cause the system to look in the wrong place [in the radar data] for the incoming Scud*

What Went Wrong?

The GAO report does not go into the technical details of the specific programming error. However, the following can be inferred based on the information and data provided about both the incident and defect.

The Code

1. The CPU was a 24-bit integer-only CPU "based on a 1970s design". Befitting the time, the code was written in assembly language.
2. Real numbers (i.e., those with fractions) were apparently manipulated as a whole number in binary in one 24-bit register plus a binary fraction in a second 24-bit register. In this fixed-point numerical system, the real number 3.25 would be represented as binary 00000000000000000000011:01000000000000000000000, in which the : is the marker for the separator between the whole and fractional portions of the real number. The first half of that binary represents the whole number 3 (i.e., bits are set for 2 and 1, the sum of which is 3). The

second portion represents the fraction 0.25 (i.e., 0/2 + 1/4 + 0/8 + …).

3. System [up]time was "kept continuously by the system's internal clock in tenths of seconds [] expressed as an integer." This is important because the fraction 1/10 cannot be perfectly represented in 24-bits of binary fraction because its binary expansion, as a series of 1 or 0 over 2^n bits, does not terminate.

The Algorithm

The missile-interception algorithm that did not work that day is understood to be approximately the following:

1. Consider each object that might be a Scud missile in the 3-D radar sweep data.
2. For each, calculate an expected next location at the known speed of a Scud (+/- an acceptable window).
3. Check the radar sweep data again at a future time to see if the object is in the location a Scud would be.
4. If it is a Scud, engage and fire missiles.

The GAO reports that the problem was an accumulating linear error of .003433 seconds per 1 hour of uptime that affected every deployed Patriot equally. This was not a clock-specific or system-specific issue.

The Tragic Result

Given all of the above, it can be reasoned that the problem was that one part of the Scud-interception calculations utilized time in its decimal representation and another used the fixed-point binary representation. When the uptime was still low, targets were found in the expected locations when they were supposed to be and the latent software bug was hidden.

Of course, all of the above detail is specific to the Patriot hardware and software design that was in use at the time of the Gulf War. As the Patriot system has since been modernized by Raytheon, many details like these will have likely changed.

According to the GAO report:

> *Army officials believed the Israeli experience was atypical [and that] other Patriot users were not running their systems for 8 or more hours at a time. However, after analyzing the Israeli data and confirming some loss in targeting accuracy, the officials made a software change which compensated for the inaccurate time calculation. This change allowed for extended run times and was included in the modified software version that was released [9 days before the Dhahran Scud incident]. However, Army officials did not use the Israeli data to determine how long the Patriot could operate before the inaccurate time calculation would render the system ineffective.*

Four days before the deadly Scud attack, the "Patriot Project Office [in Huntsville, Alabama] sent a message to Patriot users stating that very long run times could cause [targeting problems]." That was about the time of the last reboot of the Patriot missile that failed.

Note that if time samples were all in the decimal timebase or all in the binary timebase then the two compared radar samples would always be close in time and the error would not accumulate with uptime. And that is the likely fix that was implemented.

Firmware Updates

Here are a few tangentially interesting tidbits from the GAO report:

- "During the [Gulf War] the Patriot's software was modified six times."
- "Patriots had to be shut down for at least 1 to 2 hours to install each software modification."
- "Rebooting[] takes about 60 to 90 seconds" and sets the "time back to zero."
- The "[updated] software, which compensated for the inaccurate time calculation, arrived in Dhahran" the day after the deadly attack.

Public Statements

In hindsight, there are some noteworthy quotes from the 1991 news articles initially reporting on this incident. For example,

Brig. Gen. Neal, United States Command (2 days after):

> *The Scud apparently fragmented above the atmosphere, then tumbled downward. Its warhead blasted an eight-foot-wide crater into the center of the building, which is three miles from a major United States air base … Our investigation looks like this missile broke apart in flight. On this particular missile it wasn't in the parameters of where it could be attacked.*

U.S. Army Col. Garnett, Patriot Program Director (4 months after):

> *The incident was an anomaly that never showed up in thousands of hours of testing and involved an unforeseen combination of dozens of variables — including the Scud's speed, altitude and trajectory.*

Importantly, the GAO report states that, a few weeks before the Dharan Scud, Israeli soldiers reported to the U.S. Army that their Patriot had a noticeable "loss in accuracy after … 8 consecutive hours." Thus, apparently, all of this "thousands of hours" of testing involved frequent reboots. The GAO reported that "an endurance test has [since] been conducted to ensure that extended run times do not cause other system difficulties."

Note too that the quoted "thousands of hours of testing" was also misleading. The Patriot software was, also according to the GAO report, hurriedly modified in the months leading up to the Gulf War to track Scud missiles going about 2.5 times faster than the aircraft and cruise missiles it was originally designed to intercept. Improvements to the Scud-specific tracking/engagement algorithms were apparently even being made during the Gulf War.

The Need for Source Code Reviews

Once the source code was examined, these specific theories and statements about what went wrong or why it must have been a problem outside the Patriot itself were fully discredited. When computer systems may have misbehaved in a lethal manner, it is important to remember that newspaper quotes from those on the side of the designers are not scientific evidence. Indeed, the humans who offer those quotes often have conscious and/or subconscious motives and blind spots that favor them to be falsely overconfident in the computer systems. A thorough source code review takes time but is the scientific way to go about finding the root cause.

As a New York Times editorial dated 4 months after the incident explained:

> *The Pentagon initially explained that Patriot batteries had withheld their fire in the belief that Dhahran's deadly Scud had broken up in midflight. Only now does the truth about the tragedy begin to emerge: A computer software glitch shut down the Patriot's radar system, blinding Dhahran's anti-missile batteries. It's not clear why, even after Army investigators had reached this conclusion, the Pentagon perpetuated its fiction*

At least in this case, it was only a few months before the U.S. Army admitted the truth about what happened to themselves and to the public. That is to the U.S. Army's credit. Other actors in other lethal software defect cases have been far more stubborn to admit what has later become clear about their systems.

Barr Group provides testifying expert witnesses and software source code analysis teams to support complex litigation, including litigation involving product liability and infringement of intellectual property such as patents and software copyrights. CONTACT US

