Danny Crichton

CS181

Professor Eric Roberts

It's About Time: Patriot Missile Batteries and the Dangers of Software Errors

One of the major controversies of the Strategic Defense Initiative's anti-ballistic missile (ABM) shield was whether software could ever by designed to handle the incredibly difficult task of shooting down incoming missiles while in flight. David Parnas, a computer scientist, was a leading critic of the idea, writing that "because of the extreme demands on the system and our inability to test it, we will never be able to believe, with any confidence, that we have succeeded."[1] Parnas' theory would be tested just a few years later in slightly different circumstances – with deadly results.

Saddam Hussein's invasion of Kuwait in early August, 1990 prompted an immediate response from the international community. Led by the United States, an international coalition of troops was raised in the Middle East to begin a defensive, and later, offensive operation against the Iraqi regime. Unlike the insurgency facing U.S. troops in Vietnam, Iraq sported a range of technologically sophisticated weapons, including the Scud missile, developed by the Soviet Union and originally provided to Iraq as part of its war with Iran in the 1970s and 1980s.[2]

There was particular fear that Iraq may have outfitted the missiles with chemical or biological payloads – drastically increasing the risk of significant fatalities.[3] In response to the threat from these weapons, the U.S. deployed Patriot Missile batteries

[1] Parnas, pg. 44 of the Course Reader

[2] "Information Paper"

[3] Ibid.

to sensitive locations like Dhahran, Saudi Arabia, where a U.S. Army barracks was located. These defensive, surface-to-air batteries are designed to shoot down incoming enemy missiles, and they act as a sort of portable ABM shield.[4]

On February 25, 1991, Iraq launched a Scud missile at the U.S. barracks at Dhahran. Under normal conditions, the Scud should have been intercepted and eliminated by the Patriot missile batteries. However, the system failed to track the incoming missile, and 28 American troops were killed in the attack. Later investigation determined that a software error was at fault for mis-detecting the incoming missile.[5] This article looks at the technical details of this software error, the costs of the error, and cautiously looks at the people and larger forces to blame for the error.

Technical Overview[6]

The most critical function of the Patriot missile system is the unit's ability to track an incoming enemy target. Ballistic missiles like Scuds are "dumb" – they are not guided to their targets by computers or lasers, but rather take a sweeping arc from launch to destination (essentially a carefully calculated parabola). Thus, this arc gives the missiles a distinctive signature compared to other objects like planes, and the Patriot missile is designed to look for this arc in its targeting software.

To do so, the Patriot system continually makes radar sweeps of the skies, looking for metallic objects that reflect radar waves back to the transmitter. Once it has a potential target, it uses the unique characteristics of the Scud missile to predict the next location of the target. If the target falls within the predicted area (called the **range**

[4] Parsch

[5] Schmitt, Pg. 1

[6] This overview follows the outline of the GAO's report into the incident. "PATRIOT MISSILE DEFENSE."

gate), then there is a high degree of probability that the target matches a Scud missile, and the Patriot battery proceeds to shoot the target down (see Figure 1 in appendix).

The Patriot missile's computer was designed in the 1970s, and lacked the sort of high precision typical of computers today. The velocity of a detected object was saved as a floating-point decimal variable – essentially a real number. A notion of time on the machine was saved as an integer at tenth-second increments, starting from the powering up of the system. Thus, the time integer would be "10" after one second of operation, and 36,000 after one hour of continuous use.

When a target was detected, the computer needed to convert the time integer into a floating-point number for use in the prediction algorithm. To understand the precise software error under discussion, some theory of floating-point numbers must be presented. A decimal number on a computer is represented by a sequence of binary bits, which are divided into three parts – a sign bit that indicates positive or negative numbers, a significand which acts as the "value," and an exponent. To get the number from this representation, we multiply the significand by the sign bit and the exponent.

One of the major issues with floating-point numbers is that there are infinitely many real numbers, but only a finite number of possible values for a fixed-bit-length floating-point number. Thus, only some decimals have mappings to a floating-point representation, and any other real number is essentially translated to the nearest value. These "gaps" between representable numbers are closest near zero due to the exponential scaling, and they become larger the further away the number is from zero.

This conversion from the integer value of time to a floating-point number is where the primary software error took place. The registers in the Patriot's computer system (a high-speed memory used for calculations located directly on the processor) could only store 24 bits of data – lower than today's standards of 32 bits or even 64 bits. As the computer continued to operate, the value of the time integer continued to increase – and thus became harder to represent with the low-precision floating-point number due to the wider gaps at high numbers.

In many contexts, a high level of precision is not necessary. When the missile system was first developed in the 1970s, a typical cruise missile could travel as slowly as 150 miles per hour.[7] However, it proved vital in the case of the Patriot missile system. The Scud missile traveling toward Dhahran was moving at an incredible 3,750 miles per hour. Since the system had to predict the target's next location to determine the type of target, the high speed of the missile magnified any time error present in the system (as an example, a time error of .1 seconds would be off by about one-tenth of a mile). At eight hours of operation, this error was about 20% of the range gate.

Unfortunately, the system had already been operational for 100 hours at Dhahran, causing a time error of approximately .3433 seconds. After predicting the next location of the target, the Patriot system rescanned the predicted range gate for the missile. However, the high degree of error put the incoming Scud missile outside of the range gate, and thus the system failed to determine that the object was an enemy target (see Figure 2 in the appendix). It thus did not fire at the missile, and it was allowed to hit the U.S. barracks unimpeded.

[7] Kopp, all pages.

Costs

Military officers judge the cost of operations using the rubric of “blood and treasure.” The Patriot missile software error directly caused the deaths of 24 American soldiers, as well as the destruction of a significant U.S. Army barracks, likely millions of dollars in damages.

More indirectly, the Patriot missiles had to be upgraded with new software, which put the systems offline for a period of time in the middle of a war zone with active enemy attacks. While the software error was easily correctable, the loss of defensive assets in this tense context has significant ramifications.

More generally, the successful hit by Saddam’s forces decreased morale among soldiers in the international response coalition, hurting firing effectiveness. The attack was also a propaganda victory for Iraq, particularly in a war noted for its unprecedented cable media coverage. This morale issue was compounded due to the perceived risk of biological and chemical warfare. Fear increased that the United States and allied forces did not have the ability to stop Iraq’s attacks and that a “safety net” had been removed.

Responsibility: Who (or what) to blame?

Parnas’ claims regarding the Strategic Defense Initiative seem particularly prescient given the incident that happened at Dhahran. Software is in many cases complex, but the demands placed on it are extreme in the context of ABM shields. Given the loss of life and significant property damage, how should we assess blame for the incident?

The Raytheon Corporation – which produces the Patriot missile battery – would seem to be an obvious first target. The company failed to fully test the system under typical wartime operating conditions, such as using the battery for extended periods of time. Furthermore, it should have been clear that the translation system between integers and floating-point numbers was bound to fail at some point, and therefore a better operational guide to using the system should have been developed.

However, Raytheon is not an independent contractor, but a highly-dependent contractor of the Department of Defense's procurement commands. The Pentagon establishes broad controls on technology placed in the field, especially regarding reliability. Part of the reason that a computer from the 1970s is controlling 1990s missile technology is precisely because the reliability of the computer is well-known, and therefore difficult to upgrade. Thus, some blame must go to those overseeing Raytheon who failed to direct the company to test the product for longer operating times.

Similarly, the Patriot missile system gradually evolved in purpose, from anti-aircraft to cruise missiles to ballistic missiles, and the system's software was adapted to reflect these new targets. The system was designed to handle targets traveling significantly slower than Scud missiles, and thus, the software simply could not be optimized to properly handle these targets. However, the Pentagon's rules prevented Raytheon from developing new software to handle these changing tasks, and this is at least partly responsible for the Dhahran failure.

Furthermore, the dense military bureaucracy also prevents information from reaching the field as fast as might be necessary. This was the case for the Patriot missile system, in which the Pentagon was informed by Israel that the system lost reliability the longer it was operational. The Pentagon had worked to fix the problem, but the fix arrived in Dhahran on February 26, 1991 – one day too late.[8] Thus, the Pentagon is too entangled in the issue to be entirely blameless for the software.

Another partial cause of the incident was the rapid pace of events in the beginning of Operation Desert Shield. International forces were deployed within weeks of the invasion of Kuwait, and the risks of biological and chemical weapons from Scud attacks became known quite quickly as well. Given this frenetic pace, the Pentagon deployed technology to the field in many cases without fully testing it – reminiscent of the issues helicopters faced in attempting to rescue hostages in Iran at the end of the Carter presidency. Such may be the case in war, but we must be cognizant that the greater global political environment played a mediating role on the Pentagon's ability to carry out any testing program.

However, assigning blame to each of these elements – the company, the demanding client, politics – seems unfair given the state of knowledge about computers, and particularly software. The simple fact is, software can and will fail. Acceptance of this notion was a major point of Parnas' argument, but it must be extended to every person in the procurement system. The operator of the battery did not fire a Patriot missile because the software told him the incoming target was not a Scud missile – he relied on the software. Likewise, engineers at the Pentagon trusted in the ability of Raytheon to write high-quality software. The uncritical view of the power of software to solve problems is quite possibly most at fault for the incident at Dhahran – and should be the greatest lesson of this software error.

[8] "PATRIOT MISSILE DEFENSE."

Conclusion

Patriot missile systems continue to be a crucial part of America’s current anti-ballistic missile defense efforts. Perhaps unsurprisingly, the simplicity of the software error that occurred during the Gulf War hid the fundamental issue of depending on software reliability in such extreme circumstances. The software error was small – and a fix was already on the way. The deaths due to the error were just the cost of doing business. Unfortunately, the vast benefit of software obfuscates the very real danger of relying on software as a guarantee of national protection. Only by accepting a realistic picture of software’s inherent limitations will the lessons of the Dhahran incident be fully understood.

Bibliography

"Information Paper: Iraq's Scud Ballistic Missile." U.S. Department of Defense. http://www.gulflink.osd.mil/scud_info/

Kopp, Carlo. "Soviet/Russian Cruise Missiles." Technical Report APA-TR-2009-0805. Air Power Australia. http://www.ausairpower.net/APA-Rus-Cruise-Missiles.html

Parnas, David. "Software Aspects of Strategic Defense Systems." *Communications of the ACM*, Dec. 1985 Vol 28 No. 12

Parsch, Andreas. "Raytheon MIM-104 Patriot." *Directory of U.S. Military Rockets and Missiles.* 3 Dec. 2002. http://www.designation-systems.net/dusrm/m-104.html

"PATRIOT MISSILE DEFENSE: Software Problem Led to System Failure at Dhahran, Saudi Arabia" Government Accountability Office, Feb. 1992. http://archive.gao.gov/t2pbat6/145960.pdf

Schmitt, Eric. "U.S. Details Flaw in Patriot Missile." *The New York Times*. 6 Jun. 1990. http://query.nytimes.com/gst/fullpage.html?res=9D0CEED7163AF935A35755C0A967958260

Appendix: Figures

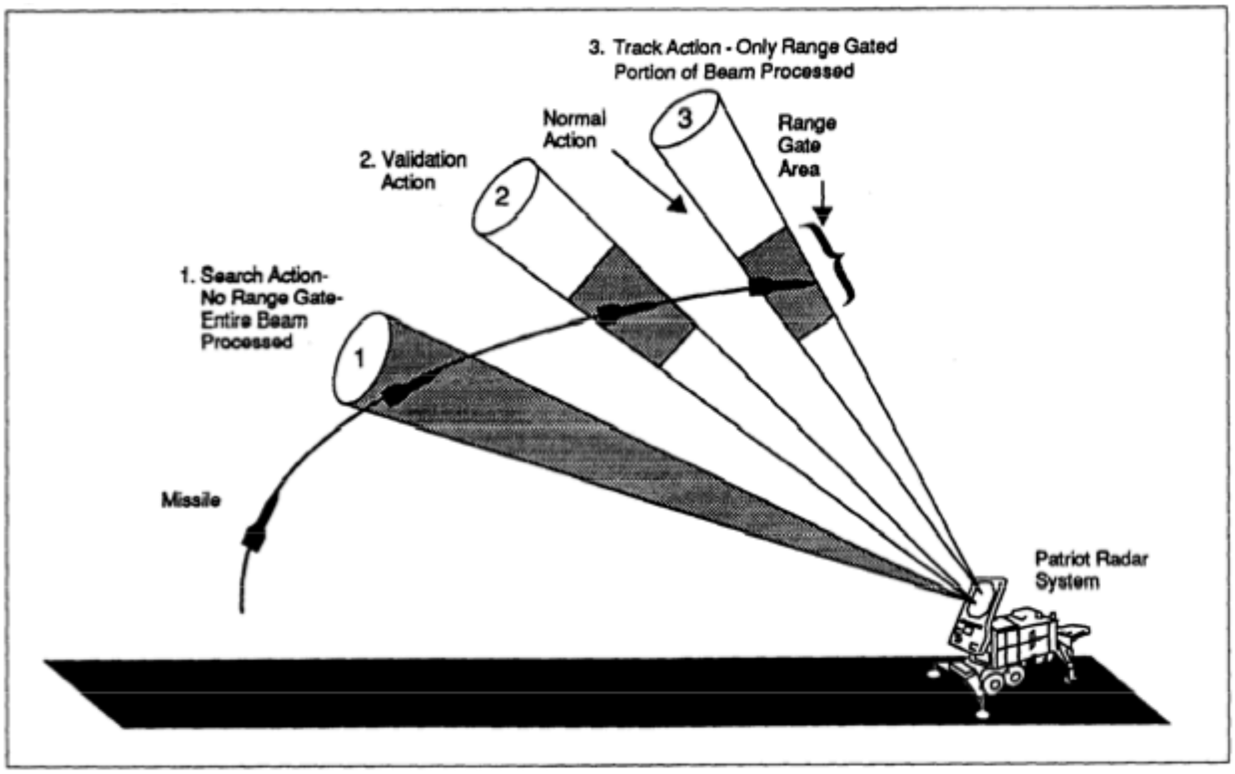
3. Track Action - Only Range Gated Portion of Beam Processed
Normal Action
Range Gate Area
2. Validation Action
1. Search Action- No Range Gate- Entire Beam Processed
3
2
1
Missile
Patriot Radar System

Figure 1: The Range Gate fully encapsulates the missile

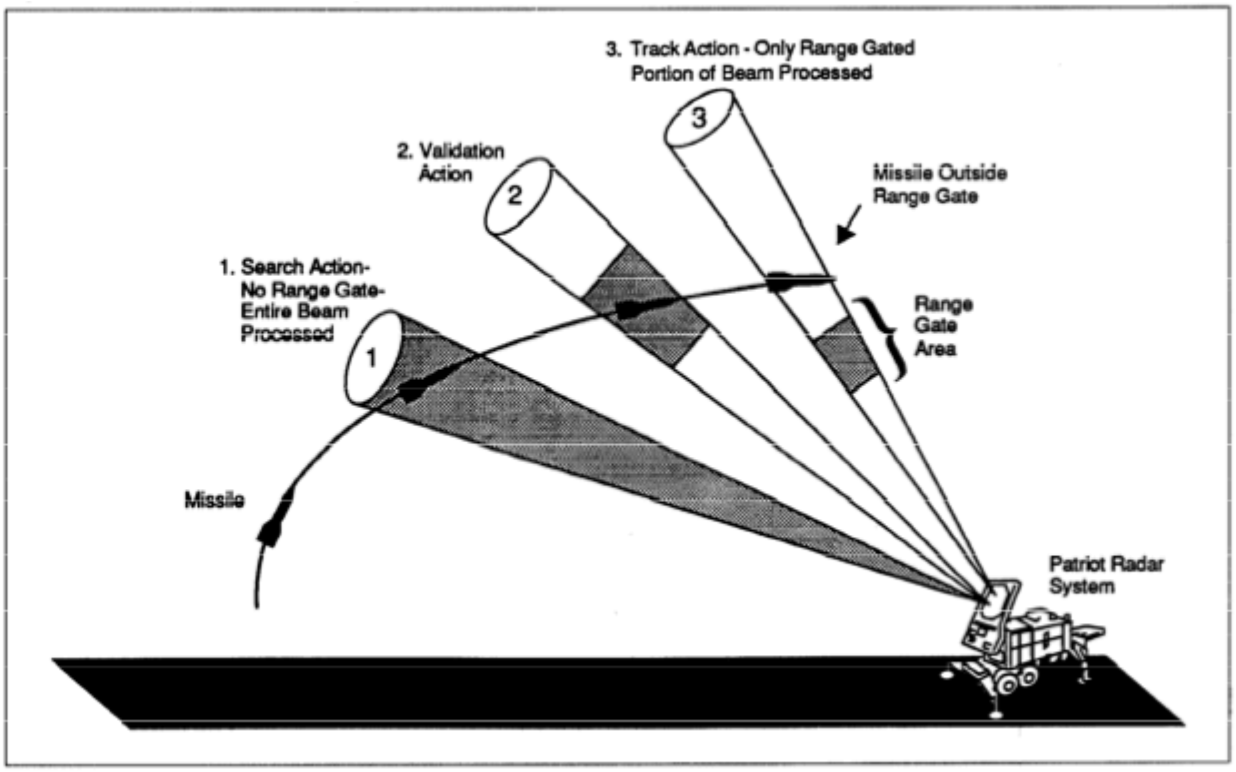
3. Track Action - Only Range Gated Portion of Beam Processed
2. Validation Action
Missile Outside Range Gate
1. Search Action- No Range Gate- Entire Beam Processed
Range Gate Area
3
2
1
Missile
Patriot Radar System

Figure 2: The Range Gate has been miscalculated and does not identify the missile