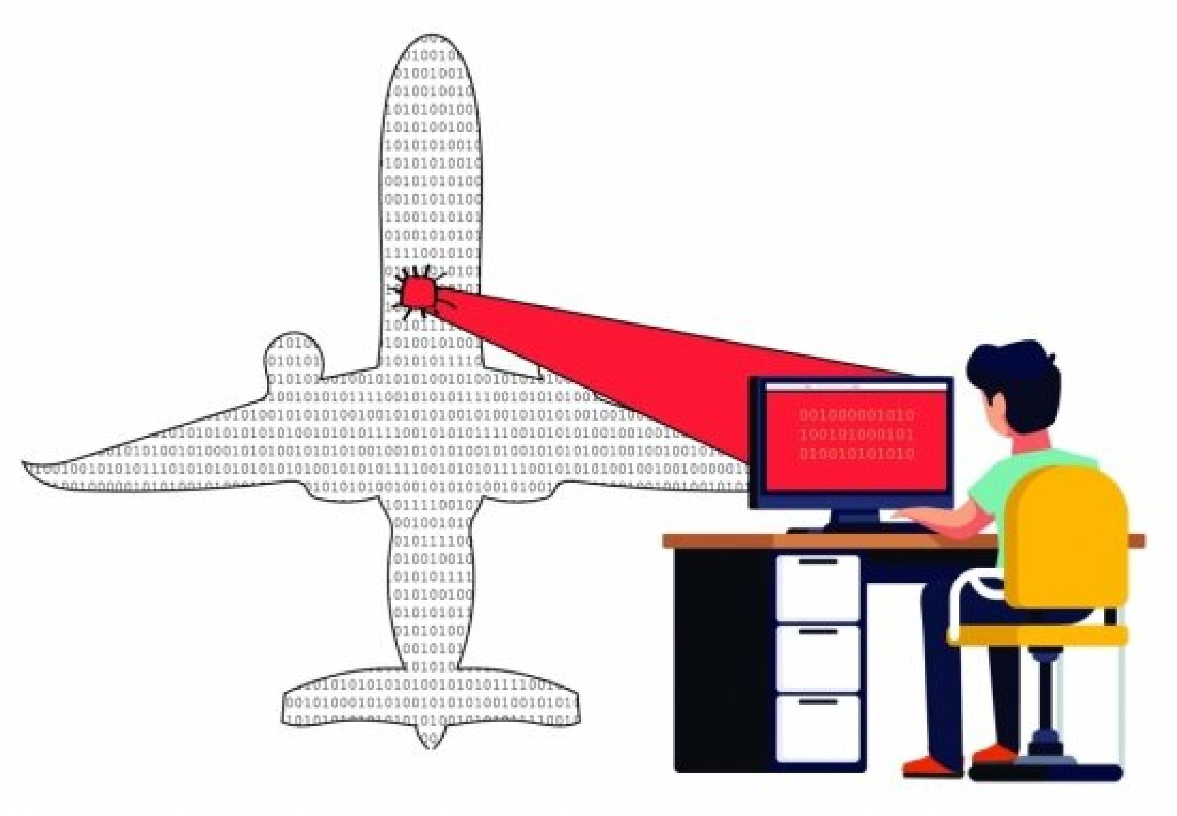
Developing a better way to address vulnerabilities at the source-code level

December 18 2020, by James Badham



Credit: LILLIAN MCKINNEY

The need to patch a problem in a program that is embedded in an existing system, perhaps to introduce or enhance security, is a common one, according to UC Santa Barbara computer science professor Giovanni Vigna. "But, why would you do that?" he asks rhetorically. "Why not just write a different program?"

"Because sometimes it's not possible, or it would require a very substantial effort," he answers. "Sometimes, you're given a program to patch, and you don't have the source code to modify it at that level and recompile it."

"Many embedded computer systems, such as those in trucks, airplanes and medical devices, run on software for which the source code and the original compilation toolchain are unavailable," said Antonio Bianchi, an assistant professor at Purdue University who studied in Vigna's lab while pursuing his doctorate at UCSB. "Many old software components running in these systems are known to contain vulnerabilities, but patching them is not always easy or even possible."

Vigna, director of the Center for CyberSecurity at UCSB, and Bianchi are among a group of researchers at UCSB, Purdue and the Swiss Federal Institute of Technology Lausanne (EPFL) who have received a four-year, $3.9 million Defense Advanced Research Projects Agency (DARPA) grant to fund a project called "Assured Micropatching," aimed at improving the process of patching code in vulnerable embedded systems.

"Without source code, patching a vulnerability necessitates editing the binary code directly," said Bianchi. "Additionally, even in a system that has been patched, there is no guarantee that the patch will not interfere with the original functionality of the device. Because of these difficulties, he said, the code running in embedded systems is often left unpatched, even when it is known to be vulnerable."

Given that situation, Vigna explained, "You need to look at the program in a different way; you need to look at the binary code, the machine code, the ones and zeroes that actually determine the behavior of a program. But until recently, binary programs were largely considered static; that is, once you had a compiled program, the thinking went, the program could not be changed. Interestingly enough, in the past, it was often only malicious actors who would focus on modifying binaries to infect benign binary programs."

But in the evolution of the field, techniques have been developed that make it possible to consider a binary program as a malleable entity, something Vigna said, "you can modify, tear apart, and put back together."

"A lot of research that has come out of UCSB," he added, "has been focused on this binary manipulation, which has a number of different applications, from protecting binaries to removing parts of a binary that you don't need or want because they may include vulnerabilities or increase the exposure to code-reuse attack, one in which part of the existing benign code is re-used to perform malicious actions."

Patching at the binary, machine-code level is extremely difficult, because there, Vigna said, "You don't have the high-level abstractions that you have in the source code, which allow you to better understand certain properties of the program."

Instead, he explained, patching the binary code is similar to fixing a house without having a blueprint or plan of the whole house, so you see only a tiny square, perhaps a door, and can only make it blue or white. "You see a tiny granular piece of it, and from there, you have to reconstruct the high-level vision of everything that's going on and then make sure that the behavior is the same," Vigna said. "You're down in the zero-one sequence, trying to figure out where the patch needs to go. This adds an enormous amount of complexity but also makes the approach applicable to everything, because everything is binary. Your heater, your microwave oven—they are all controlled by microprocessors running binary code."

Naturally, computer scientists want to avoid having to recompile an entire system to fix a single vulnerability. "I just want to be able to make this change to this binary that will fix a probable vulnerability and not have to think about it anymore," Vigna said. "And I want some formal assurance that I didn't make a mistake and introduce some new vulnerabilities, or create an unstable program. A lot of work goes into being able to demonstrate to you that my modification has fixed the problem without changing the behavior of the program."

Another key to a desirable patching system is that it need not be retested once the patch is made. "If I can prove to you that the vulnerability is fixed but that the system will otherwise operate in exactly the same way as it always did, then you don't have to test it again," Vigna noted. "That saves time, improves security and employs the system that makes binary code malleable."

Vigna, working with fellow computer science professor Christopher Kruegel and with Aravind Machiry, newly minted Ph.D. from UCSB and soon-to-be professor at Purdue, has developed new technology to verify security patches, but the proposed technique applies only to source code. The new DARPA project is focused on extending the approach to binary code to improve its effectiveness and expand its applicability to the software running on the millions of devices that surround us every day.

Provided by University of California - Santa Barbara

Citation: Developing a better way to address vulnerabilities at the source-code level (2020, December 18) retrieved 27 April 2024 from https://techxplore.com/news/2020-12-vulnerabilities-source-code.html