BILL WONG | Embedded/Systems/Software Editor
bill.wong@penton.com

Are You Writing
Safe And Secure Software?

Secure Boot, iPhones, mulch, safety, and Ada—I promise to tie them all together, since they're all related. Sort of.

As a programmer, I try to write code that's not only functional but also safe, secure, and reliable. I work at learning the best techniques, using the best compiler tools, and taking advantage of the latest technology. Great. Everyone knows that C dominates the embedded space. But what if some of those things I learn or use aren't quite right?

Georgia Tech students attacked a poor, helpless iPhone using a string overflow bug in the phone's USB charger DRM support designed to restrict chargers to those licensed by Apple.

... BUT EVERYONE USES MULCH

So now the mulch. You've seen it everywhere on landscaped lawns. There are giant mounds around every tree. It looks great, if you like that kind of thing. It makes mowing, especially with large riding lawnmowers, so easy.

Unfortunately, piling bark mulch around most anything, especially trees, isn't a great idea. Compost is good around bushes, but trees don't like anything stacked up on their bark. There are other good uses for bark mulch, though.

C and C++ aren't quite as bad as mulch around trees, but using them just because everyone else does can be an issue because they're prone to bugs. Dangling pointers and buffer overflows are common C bugs.

... AND NOW THE IPHONE

Researchers at my alma mater, Georgia Tech, attacked an iPhone using a smart USB charger *(see the figure)*. Apple decided to use digital rights management (DRM) to restrict licensing of high-current USB chargers. Essentially, the charger handshakes with the phone using the USB data channel before it raises the amount of power delivered.

The charger sends a key in its handshake packet. The researchers caused a buffer overflow. This form of injection attack is common in jailbreaking smart phones and other devices. They take advantage of a software bug.

The number of buffer overflow attacks on everything from smart phones to servers is rather extensive, though. One might think that avoiding these kinds of problems would improve the safety and security of quite a few systems.

The new USB standard also provides a high-current feature, but it does not use the data link. Instead, it uses some creative analog signalling.

READ ABOUT ADA 2012, TO BE SAFE

This brings me to Ada, which is designed from the ground up for safety and security. Its features also turn out to make bug-free programming much easier. But from a programming perspective, I'm still a neophyte. I have been using it more and learning quite a bit about it, but I defer to the experts in trying to convince programmers about its advantages.

Safe and Secure Software - An Invitation to Ada 2012, a free e-book from AdaCore available at *www.adacore.com/safe-secure-booklet*, highlights programming issues that arise with any programming language. Of course, it explains why Ada 2012 is a better solution often because it does things that other languages, including C++ and Java, do not address.

For example, Ada's type system puts most others to shame. Its pointer support provides finer-grain control, allowing the compiler to catch more errors. Ada's storage management is something that most languages relegate to libraries, but there are reasons for incorporating it into the compiler.

Ada 2012 also brings contracts out of the SPARK comment realm and into the language itself. This is reason enough to take another look at Ada, but there is plenty more. The book shows the problems, how they are addressed with other languages, and why Ada was designed to address these issues.

The book offers a flavor of Ada, but it is designed for any programmer. It addresses Ada basics like built-in range checking that probably would have caught the iPhone charger problem. Hopefully, the book will get you thinking.

Still, C and C++ won't be abandoned en masse. Tools like MISRA C can help. I recommend looking into static and dynamic analysis tools because they do make a difference. ed

Follow Bill Wong on Twitter at #AltEmbedded.

Become a fan at facebook.com/ElectronicDesign.