In this final column in a short series dealing with security in programming, my manager at AtomicTangerine, Mike Gerdes, contributed the following suggestions and comments.

* Might I suggest that you recommend the readers adopt a practice of designing code in a more holistic fashion? A common practice is to write and test routines in a way that verifies the code processes the data in the way intended. To avoid the effects of malicious code and data input attacks, the programmer must also write code which deals with what is NOT supposed to be processed. A more complete design methodology would also include testing of all inbound information to ensure exclusion of any data which did not fit the requirements for acceptable data. This method should be applied to high risk applications and those with an extremely arduous test cycle and will eliminate many of the common attack methods used today.

* Establish the criteria for determining the sensitivity level of information contained in, or processed by the application and subroutines.

* If they are not already present, consider implementing formal control procedures in the software programming methodology to ensure all data is reviewed during QA processes to be sure it is classified and handled appropriately for the level assigned.

* Identify and include any mandatory operating system and network security characteristics for the production system in the specifications of the software. In addition to providing the development and QA teams some definition of the environment the software is designed to run in, giving the administrator and end users an idea of what your expectations were when you created the code can be extremely useful in determining where software can, or cannot, be used.

* Where appropriate, verify the digital signatures of routines that process sensitive data when the code is being loaded for execution.

* If you include checksums on executables for production code, include routines which verify the checksums at every system restart.

* * *

In addition to my thanks to Mike Gerdes for the ideas included above, I thank our friend and colleague Edwin Blackwell at AtomicTangerine for his helpful comments on the original text of these articles. I look forward to suggestions from readers (non-programmer readers might want to circulate the articles to their programming colleagues for ideas) on how to expand and improve this list of suggestions as well as suggestions on good books and URLs dealing with security in programming, all to be published in a followup article later. Contributors should tell me if I can credit you by name and affiliation in a column.

* * *
Mich Kabay can be reached by e-mail at <mkabay@atomictangerine.com>. He invites inquiries about a wide range of information security courses and INFOSEC consulting services that he and his colleagues at AtomicTangerine would be delighted to deliver to your employees at your site and at your convenience. For Web-based or CD-ROM online training in security from our INFOSEC University project, see <http://infosecu.com>.

For information about AtomicTangerine, visit <http://www.atomictangerine.com>.

Copyright © 2001 M. E. Kabay. All rights reserved.

Permission is hereby granted to Network World to distribute this article at will, to post it without limit on any Web site, and to republish it in any way they see fit.