Foreword

Society's increased dependency on networked software systems has been matched by an increase in the number of attacks aimed at these systems. These attacks—directed at governments, corporations, educational institutions, and individuals—have resulted in loss and compromise of sensitive data, system damage, lost productivity, and financial loss.

While many of the attacks on the Internet today are merely a nuisance, there is growing evidence that criminals, terrorists, and other malicious actors view vulnerabilities in software systems as a tool to reach their goals. Today, software vulnerabilities are being discovered at the rate of over 4,000 per year. These vulnerabilities are caused by software designs and implementations that do not adequately protect systems and by development practices that do not focus sufficiently on eliminating implementation defects that result in security flaws.

While vulnerabilities have increased, there has been a steady advance in the sophistication and effectiveness of attacks. Intruders quickly develop exploit scripts for vulnerabilities discovered in products. They then use these scripts to compromise computers, as well as share these scripts so that other attackers can use them. These scripts are combined with programs that automatically scan the network for vulnerable systems, attack them, compromise them, and use them to spread the attack even further.

With the large number of vulnerabilities being discovered each year, administrators are increasingly overwhelmed with patching existing systems. Patches can be difficult to apply and might have unexpected side effects. After

a vendor releases a security patch it can take months, or even years, before 90 to 95 percent of the vulnerable computers are fixed.

Internet users have relied heavily on the ability of the Internet community as a whole to react quickly enough to security attacks to ensure that damage is minimized and attacks are quickly defeated. Today, however, it is clear that we are reaching the limits of effectiveness of our reactive solutions. While individual response organizations are all working hard to streamline and automate their procedures, the number of vulnerabilities in commercial software products is now at a level where it is virtually impossible for any but the best resourced organizations to keep up with the vulnerability fixes.

There is little evidence of improvement in the security of most products; many software developers do not understand the lessons learned about the causes of vulnerabilities or apply adequate mitigation strategies. This is evidenced by the fact that the CERT/CC continues to see the same types of vulnerabilities in newer versions of products that we saw in earlier versions.

These factors, taken together, indicate that we can expect many attacks to cause significant economic losses and service disruptions within even the best response times that we can realistically hope to achieve.

Aggressive, coordinated response continues to be necessary, but we must also build more secure systems that are not as easily compromised.

About *Secure Coding in C and C++*

Secure Coding in C and C++ addresses fundamental programming errors in C and C++ that have led to the most common, dangerous, and disruptive software vulnerabilities recorded since CERT was founded in 1988. This book does an excellent job of providing an in-depth engineering analysis of programming errors that have led to these vulnerabilities and mitigation strategies that can be effectively and pragmatically applied to reduce or eliminate the risk of exploitation.

I have worked with Robert since he first joined the SEI in April 1987. Robert is a skilled and knowledgeable software engineer who has proven himself adept at detailed software vulnerability analysis and in communicating his observations and discoveries. As a result, this book provides a meticulous treatment of the most common problems faced by software developers and provides practical solutions. Robert's extensive background in software development has also made him sensitive to tradeoffs in performance, usability, and other quality attributes that must be balanced when developing secure code. In addition to

Robert's abilities, this book also represents the knowledge collected and distilled by CERT operations and the exceptional work of the CERT/CC vulnerability analysis team, the CERT operations staff, and the editorial and support staff of the Software Engineering Institute.

—Richard D. Pethia
CERT Director