



Targeting the 25 Most Dangerous Programming Errors

It may be hard to believe, but many computer security breaches are the result of programming errors that could easily have been prevented had developers known the risk. Last year, hackers took advantage of two programming errors to modify hundreds of thousands of trusted Web pages. “The attack worked because countless programmers made the exact same mistake in their software,” says MITRE’s Steve Christey, a principal information security engineer. “In 2005, a teenager exploited these same two errors and created a worm that affected over 1 million MySpace users in less than a day, causing a temporary outage for the entire site.”

Now, a list of the most dangerous programming errors can help businesses and government stop such attacks and improve the security of their software. It’s called the 2009 Common Weakness Enumeration/SANS Top 25 Most Dangerous Programming Errors. The Top 25 list is the result of collaboration among MITRE; the SANS Institute, a leading information security and education company; and top software experts in the U.S. and Europe.

The Top 25 list spells out the most significant programming errors that can lead to serious software vulnerabilities. “These errors occur frequently, are often easy to find, and easy to exploit,” says MITRE’s Robert A. Martin, a principal engineer and CWE program manager. “They’re dangerous because they’ll frequently allow attackers to completely take over the software, steal data, or prevent the software from working at all.”

Top 25 Derived from Common Weakness Enumeration Project

The Top 25 is drawn from more than 700 errors listed in the Common Weakness Enumeration (CWE) program that MITRE has developed in collaboration with the software security community since 2005. CWE is sponsored by the Department of Homeland Security’s National Cyber Security Division (NCSD) Software Assurance Program. The National Institute of Standards and Technology (NIST) also participates in CWE through the Software Assurance Metrics and Tool Evaluation project, which is sponsored by the DHS NCSD Software Assurance Program. Through that effort, NIST collaborates with MITRE and tool vendors and leverages CWE to provide a common basis for evaluating static analysis tools.

Martin defines a weakness as something that can go wrong in your software’s architecture, design, code, or the environment it’s running in. “Weaknesses are the root cause behind all the patches we have to make,” he says. “Someone made a mistake in the architecture, design, or coding implementation. But it’s not just people who offer commercial or open-source software that make these mistakes. Everyone who writes software—whether it’s for sale, sharing, or internal use—can let these weaknesses slip in. Therefore, they need to understand how they can happen and how to avoid them.”

Martin points out that both the Top 25 site and the CWE site have “a lot of information on how to prevent and mitigate these weaknesses. There are examples of both good code and bad code so you can understand how a mistake can be made. And understanding how these weaknesses can be attacked is an important aspect of dealing with them.”



Many computer breaches are caused by programming errors that could easily have been prevented.

These errors...are dangerous because they’ll frequently allow hackers to completely take over the software, steal data, or prevent the software from working at all.

Improper input validation “is the number one killer of healthy software...”

Based on Frequency and Severity

The list was built by estimating how frequently these weaknesses appear and how severe the damage could be. The Top 25 list is grouped into three categories:

1. Insecure interaction between components
2. Risky resource management
3. Porous defenses

The first category includes errors such as improper input validation. “It’s the number one killer of healthy software, so you’re just asking for trouble if you don’t ensure that your input conforms to expectations,” says Christey.

Why do security errors get introduced? “In the past, software developers weren’t educated about the hazards of programming errors,” says Martin. “It was a safer world and computer networks were isolated from each other. But today, the environment has changed, and everything is accessible. A handful of universities around the country now teach software security. For the most part, however, software developers acquire their awareness about security on the job rather than through a targeted program.”

Four Major Impacts

The Top 25 is expected over time to make an impact in four major areas. Software buyers will be able to buy much safer software. Programmers will have tools that consistently measure the security of the software they are writing. Colleges will be able to teach secure coding more confidently. And employers will be able to ensure they have programmers who can write more secure code.



The Top 25 list is drawn from over 700 errors, or weaknesses, compiled since 2005 by MITRE’s Bob Martin, left, and Steve Christey.

Interest in programming errors and how to prevent them is high. The day after the Top 25 list was announced, the number of hits on MITRE’s Common Weakness Enumeration site (cwe.mitre.org) went from 306 to more than 72,000.

The Office of the Director of National Intelligence expressed its support saying, “We believe that integrity of hardware and software products is a critical element of cybersecurity. Creating more secure software is a fundamental aspect of system and network security, given that the federal government and the nation’s critical infrastructure depend on commercial products for

business operations. The Top 25 is an important component of an overall security initiative for our country. We applaud this effort and encourage the utility of this tool through other venues such as cyber education.”

Martin notes that “the list has already had an impact on the procurement practices of the State of New York. The Top 25 is mentioned specifically in the section about ‘Vulnerabilities, Risks, and Threats.’ New York State basically wants a guarantee from a software vendor that the state is protected against the Top 25.” The specific language says: “The Vendor shall conduct an analysis of the attached 25 most common programming errors and document in writing that they have been mitigated.”

Christey points out that the Top 25 is “not an end in itself. It’s the first step in a continuing process.” Plans call for the list to be updated annually. And while the errors included will vary from year to year, the sharing of the information no doubt will put a crimp in hackers’ ability to wreak havoc on computer networks.

—by David A. Van Cleave

Contact: For more information on this and other MITRE programs, see www.mitre.org/news/digest

©2009 The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution unlimited; Case number: 09-0301