Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

Developing Secure Software

New York City SPIN
May 11, 2004

Noopur Davis
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Sponsored by the U.S. Department of Defense

Outline

Defective Software is not Secure

The Response Strategy

The Secure Software Development Strategy

Defective Software is not Secure

Common software defects are a principal cause of software vulnerabilities.

- Over 90% of software security incidents are due to attackers exploiting known software defect types.
- Top 10 causes account for about 75% of all vulnerabilities.

Source: CERT® Coordination Center (CERT/CC)

What is a Vulnerability?

Different people have different definitions.

The CERT/CC has a shared understanding

- violates an explicit or implicit security policy
- usually caused by a software defect
- similar defects are classified as the same vulnerability
- often causes unexpected behavior

We specifically exclude from “vulnerability”

- Trojan horse programs (evil email attachments)
- viruses and worms (self propagating code)
- intruder tools (scanners, rootkits, etc.)

Vulnerabilities are the defects that permit these things to exist.

Security Defects

Examples:

- failure to authorize and authenticate users
- failure to encrypt and/or protect sensitive data
- improper error handling
- improper session management

Everyday software “bugs” are also a major risk.

For example, a buffer overflow can cause system failure or allow a hacker to take control of a system.

Many common defect types can produce a buffer overflow

- declaration error
- logic errors in loop control or conditional expression
- failure to validate input
- interface specification error

The Response Strategy

The current software security approach could be called a response strategy.

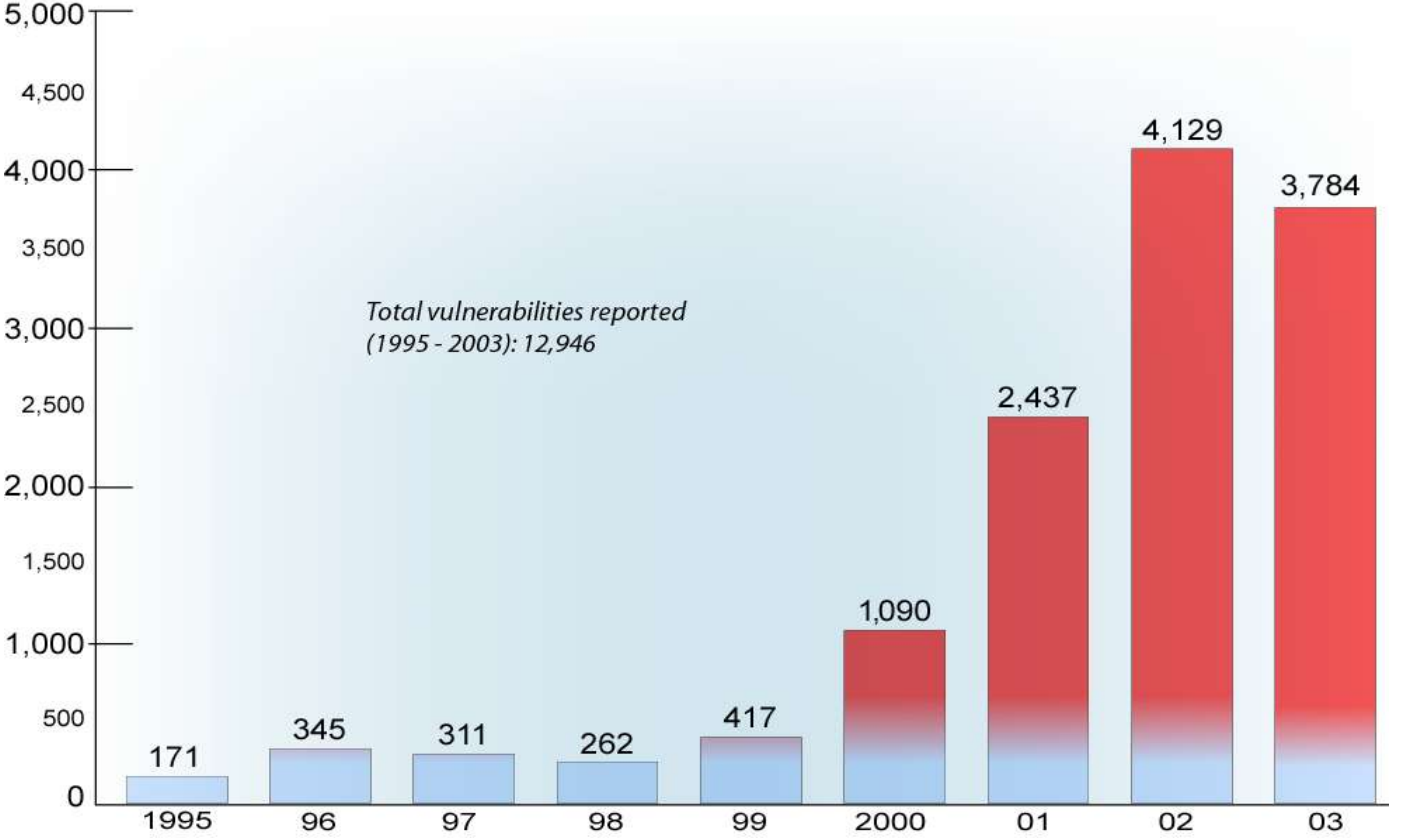
The development of software for secure applications is handled the same way as other software.

This typically results in many delivered defects.

The manufacturer then waits for attackers to find vulnerabilities before developing fixes.

The system's users then apply these fixes to prevent further similar attacks.

There Are Too Many Vuls to Patch



The Administrative Cost

- The BITS organization estimates software vulnerabilities cost BITS and Financial Services Roundtable members $400 million annually and the financial sector in total more than $1 billion.
- Intel applied 2.4 million patches to its own network.
- A scan of 470 machines reported that 8,000 patches needed to be applied.
- An organization with 100,000 IP addresses could be subjected to 2.3 million vulnerability probes per day.
- Aberdeen Group estimates the cost to U.S. businesses to manage security vulnerabilities was $3.5 billion in 2002.

Sources: BITS, Intel white paper, press, Aberdeen Group, CERT/CC

The Response Strategy Is Failing

The response strategy accepts the costs of initial attacks.

It is impractical for system administrators.

It is expensive for suppliers.

- excessive development and repair costs
- unknown and possibly unlimited litigation exposures

The response strategy cannot consistently or economically produce secure systems.

The Secure Software Development Strategy

Problem

- Testing is not enough.
- Inspections and reviews are not enough.
- Use of tools is not enough.
- Design principles are not enough.
- Risk management is not enough.

First, there is the need for education.

Second, there is a need for a process that combines all of the above in a planned, managed, and measured framework.

- Uses outstanding software engineering practices that produce near defect-free software.
- Incorporates best security practices.
- Is supported by best management practices.
- Uses measurement to judge effectiveness and continuously improve.

TSP(SM) For Secure Software Development

Research objectives

- Reduce or eliminate software vulnerabilities that result from software design and implementation defects.
- Provide the capability to predict the likelihood of latent vulnerabilities in delivered software.

Areas of exploration

- Vulnerability analysis by defect type
- Operational process for secure programming
- Predictive process metrics and checkpoints
- Quality management practices for secure programming
- Design patterns for common vulnerabilities
- Verification techniques
- Removing vulnerabilities in legacy systems

Summary

Defective software is not secure.

Software quality is a pre-requisite for secure software.

A secure software development process

- Is based on outstanding software engineering practices that produce near defect-free software.
- Incorporates best security practices.
- Is supported by best management practices.
- Uses measurement to judge effectiveness and continuously improve.

Contact Information

For more information about collaboration:

Robert Rosenstein

Business Manager
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
412-268-8468 – Phone
412-291-3054 – FAX
412-818-3446 – Mobile
br@sei.cmu.edu - Email