Insecurity in Security Software

Maik Morgenstern
Andreas Marx
AV-Test GmbH

http://www.av-test.org

Virus Bulletin 2005 Conference presentation about “Insecurity in Security Software”

Phone: +49 391 6075460, Fax: +49 391 6075469, http://www.av-test.org

Table of content

- The paradox
- Types of security software
- Comparison of CVE advisories
- Examples of bugs and security vulnerabilities
- Why bugs occur
- Vulnerability lifecycle
- What to do? (for users and developers)
- Trustworthy computing development lifecycle

The paradox

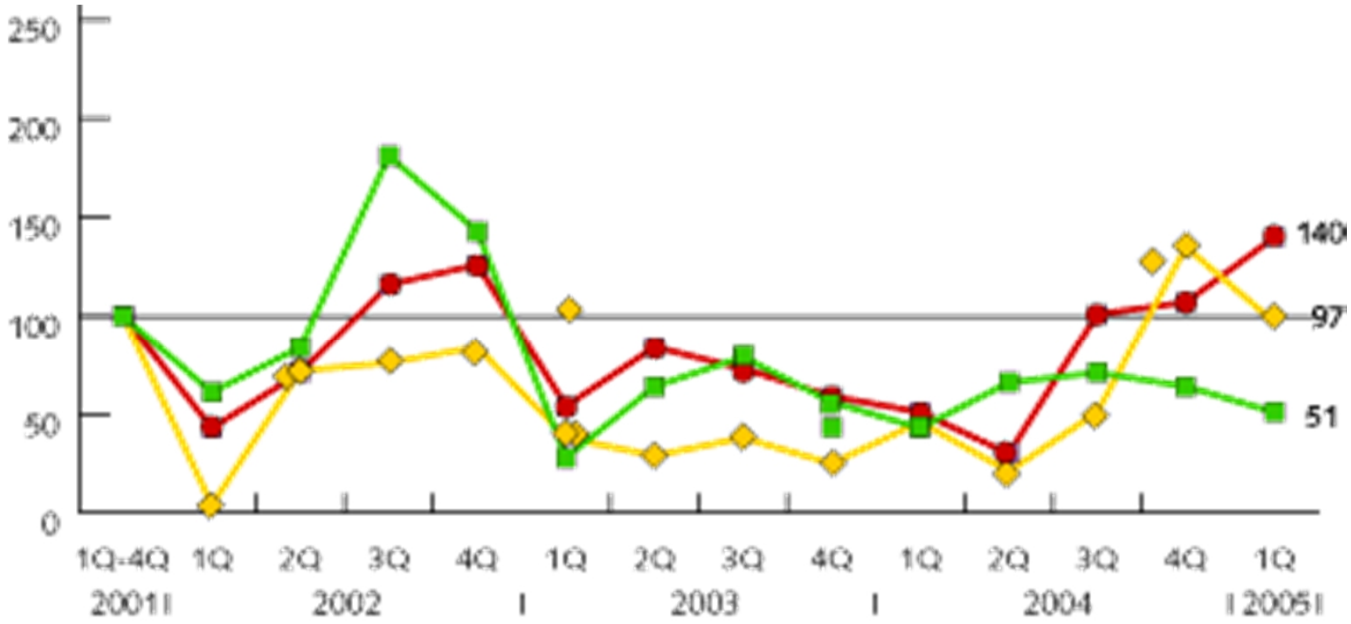
- All software products contains security vulnerabilities (and other bugs)
- AV software is widely deployed to protect companies, organizations and home users
- Every week, security flaws are discovered in different AV products
- The paradox: Security software is meant to secure the system, but nowadays it introduces new security holes.

Types of security software

- Two different groups of security software:
  - Home and business user software (widely used)
    - Firewalls
    - IPSec products
    - IDS/IPS
    - AV software…
  - Tools used by researchers (small deployment)
    - IDA Pro
    - OllyDbg
    - Softice…

CVE advisories for vendor products

(2001 quarterly average = 100, Source: © The Yankee Group)



- **Microsoft / Security vendors / All vendors**

Bug leading to a security vulnerability

- A couple of examples from the last months (advisory titles):
  - ISS and the Witty Worm
  - Trend Micro VSAPI ARJ parsing
  - McAfee Virus Library
  - Symantec Multiple Products UPX Parsing Engine Heap Overflow
  - Computer Associates Vet Antivirus Library Remote Heap Overflow
  - Kaspersky AntiVirus "klif.sys" Privilege Escalation Vulnerability
  - OllyDbg "INT3 AT" Format String Vulnerability
  - DataRescue IDA Pro Dynamic Link Library Format String Vulnerability
  - Clam AntiVirus ClamAV Cabinet File Handling DoS Vulnerability

Bugs vs. security vulnerabilities

- Some more examples of the last months:
  - Trend Micro Virus Sig 594 causes systems to experience high CPU utilization
  - Windows NTFS Alternate Data Streams
  - Archive Problems
  - BitDefender bug bites GFI
  - Panda AntiVirus deleting Tobit David communications software
  - Symantec Brightmail AntiSpam Static Database Password
  - McAfee Internet Security Suite 2005 Insecure File Permission

Why bugs occur: 3 main factors

- Technical factors
  - The underlying complexity of the task itself
- Psychological factors
  - The “mental models,” for example, that make it hard for human beings to design and implement secure software
- Real-world factors
  - Economic and other social factors that work against security quality

- Source: Mark G. Graff, Kenneth R. van Wyk, ‘Secure Coding: Principles & Practices’, O'Reilly, 2003

Vulnerability lifecycle

- A never-ending story!
  1. Discover vulnerability
  2. Develop patch
  3. Get alert and install patch
  4. Goto 1

- Source: Mark G. Graff, Kenneth R. van Wyk, ‘Secure Coding: Principles & Practices’, O'Reilly, 2003

What to do? (I)

- Corporate users:
  - Update your products frequently!
  - … not only signature files in case of AV software, but really all components (e.g. engine, GUI)!
  - Read publicly available information about newly discovered flaws and don‘t call the people first
  - Try to shorten test intervals (months vs. weeks) for security vulnerability related updates
  - “Scan throughput” is not the only important thing!

What to do? (II)

- Software developers:
  - Check your old “known-working” code
  - Check for updates of 3rd party software included in your products
  - File format “Sandbox” (enforce protocol)
  - Strategy to use minimal rights only whenever possible (do not use Administrator or Root rights)
  - Create easy update deployment mechanisms

Trustworthy computing development lifecycle (I)

- Four principles of secure development:
  - Secure by design
  - Secure by default
  - Secure in deployment
  - Communications

- Source: Steve Lipner, Michael Howard, ‘The Trustworthy Computing Security Development Lifecycle’, Microsoft 2005

Trustworthy computing development lifecycle (II)

- Example (Microsoft‘s suggestions):
  - Implementation phase:
    - Apply coding and testing standards
    - Apply security-testing tools including fuzzy logic
    - Apply static analysis code scanning tools
    - Conduct code reviews

- Source: Steve Lipner, Michael Howard, ‘The Trustworthy Computing Security Development Lifecycle’, Microsoft 2005

Summary

- Security vulnerabilities are an industry-wide problem
- Microsoft isn’t the only target today anymore
- Every error could be security relevant when it happens in security software!
- Proactive actions (e.g. automated and manual code reviews, rewriting of code) has to be considered
- Implement several layers of security (“Sandbox”)
- Responsible way of updating: “Update often, update early, not too often and not too early”

Any questions?

- Are there any questions?