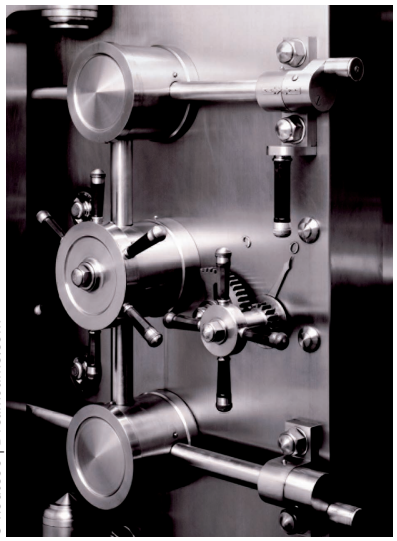

© Route66 | Dreamstime.com

Vulnerability Trends: Measuring Progress

Rick Kuhn and Chris Johnson, *US National Institute of Standards and Technology*

The first installment of this column, back in January 2009, reviewed trends in software vulnerabilities.[1] Now that roughly one Moore's Law generation has passed, it seems appropriate to revisit vulnerability trends.

Interesting News

We analyzed data from the National Vulnerability Database (NVD). Designed and operated by the National Institute of Standards and Technology (NIST) with support from the Department of Homeland Security, the NVD provides fine-grained search capabilities of all publicly reported software vulnerabilities since 1997—a total of 41,810 vulnerabilities for more than 20,000 products. Frequently, a single vulnerability can affect a large number of products—for example, when the fault occurs in a library function.

A note on the data limitations—our discussion in this article is based on data in the NVD, which relies on publicly reported vulnerabilities from the Common Vulnerabilities and Exposures (CVE) dictionary. Not all products and vulnerabilities are included in the NVD data. For example, products with low usage might not be included. Also, software vendors may change their vulnerability reporting practices over time, such as no longer reporting certain classes of vulnerabilities that they deem relatively unimportant. Readers should be aware that the trends demonstrated in this column are based on partial data.

Additionally, older data might not be as accurate; the NVD adopted version 2.0 of the Common Vulnerability Scoring System (CVSS) in June 2007, and most vulnerabilities prior to this date were natively scored using the version 1.0 guidelines and subsequently converted to an approximated version 2.0 score. Readers should keep in mind these limitations when interpreting the data.

As we will see, the news isn't bad, despite increasingly sophisticated attackers. At one time, the typical attacker was most likely a petty criminal or cracker, but today's systems are targeted by organizations with significant resources. Software can be vulnerable not only because of design or implementation errors but also because people have discovered better attacks, just as armor that offers protection from small arms fire can be penetrated by more powerful munitions. Eliminating flaws that lead to vulnerabilities is vital for IT systems, but making those that remain less potentially damaging and harder to exploit can improve security as well.

Vulnerability Severity

Figure 1 shows that some progress is being made. Since 2006, vulnerabilities have declined by 26 percent, despite the ever-growing number of applications. (It's important to note that figures for 2010 are projected based on four months, January to April. NVD data varies little by quarter and is approximately normally distributed with a standard deviation of three percentage points.)

Figure 1 distinguishes between high-, medium-, and low-severity vulnerabilities, based on the CVSS,[2] which assigns a numeric composite score that considers the impact on confidentiality, integrity, and availability. Impact for these three aspects of security can be none, partial, or complete, and scores combine these impacts into a single number. Essentially, a low score means there was limited adverse effect on the organization, medium indicates a serious

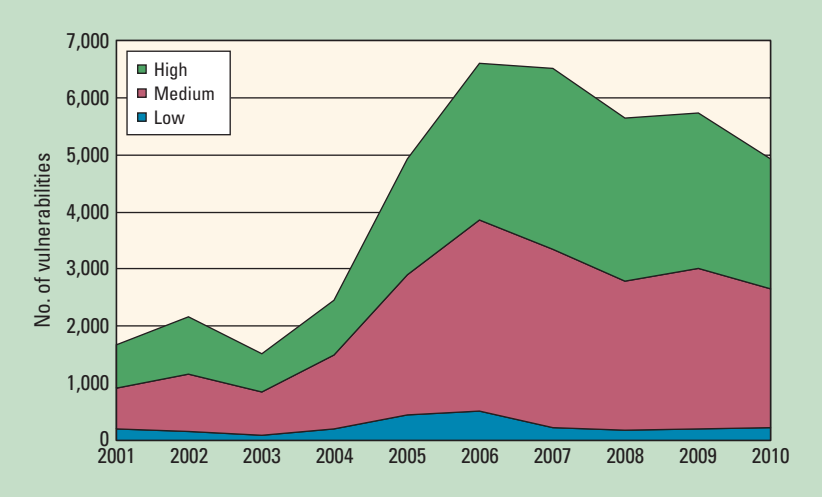

Figure 1. Vulnerabilities by severity. Vulnerabilities have declined approximately 26 percent since 2006, but the proportion of high, medium, and low severity has remained relatively constant.

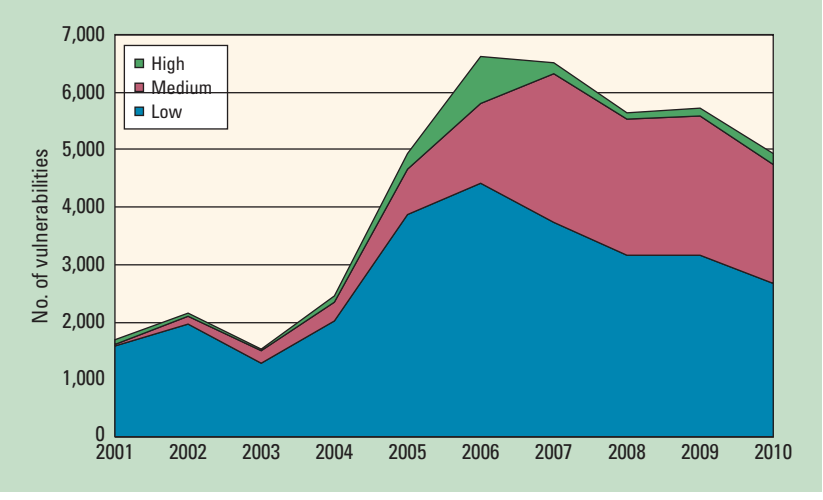

Figure 2. Vulnerabilities by access complexity. Before 2006, almost all vulnerabilities were easy to exploit.

adverse effect, and high is considered catastrophic.

Although reported vulnerabilities have been declining, it's apparent from the data that the proportion at each severity level has changed relatively little in the past 10 years.

Access Complexity

In Figure 2, we see additional progress over the past decade. Until 2006, vulnerabilities rated as low in terms of access complexity tracked closely with the total—in other words, almost all vulnerabilities were easy to exploit.

For this component of the CVSS, a low attack complexity means one that involves no specialized conditions, such as a default configuration or an attack that can be conducted manually and requires little skill. (As a conservative measure, the low complexity totals include cases where there isn't sufficient information to assign a category. This might occur when the mechanics for exploiting a vulnerability aren't well understood or when a vendor doesn't fully disclose detailed information for a vulnerability.)

Medium complexity means that access conditions are somewhat specialized—for example, involving nondefault conditions or requiring specific system knowledge in advance.

High complexity refers to specialized access conditions, such as rarely seen configurations or race conditions with a narrow window.

Since 2006, low access-complexity vulnerabilities have dropped as a percentage of the total.

Access Vector

The IT environment has become increasingly complex in the past 10 years, primarily with the growth of Internet commerce. The number of Web servers on the Internet has increased from roughly 26 million in 2001, to 74 million in 2006, to 205 million as of April 2010.[3]

As reflected in the green band of Figure 3, the number of vulnerabilities that are locally exploitable has fluctuated around an annual average of approximately 500 for the 10-year period, while network-based vulnerabilities have increased by a factor of four or more.

In this case, *local* access means either physical access to the machine or availability of a shell. *Network access*, often referred to as "remotely exploitable," means that an attack doesn't require local or local-network access. An *adjacent network* refers to a local network such as a TCP/IP subnet or a wireless or Bluetooth network. Adjacent-network vulnerabilities are too few to be visible in Figure 3 but are detailed in Table 1.

It's interesting that adjacent-network vulnerabilities peaked in 2007 and now appear to be

Table 1. Adjacent-network vulnerabilities remain a small component.

Vulnerability	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
Network	1,193	1,754	1,194	2,028	4,140	6,012	5,887	5,037	5,270	4,518
Adjacent network	0	0	2	0	4	18	38	10	16	0
Local	484	402	331	424	788	578	589	585	447	405

declining, suggesting that developers are implementing appropriate controls when integrating these technologies into applications. In today's increasingly networked world, the overall reduction in vulnerabilities of the past few years is encouraging.

Room for Improvement

Despite the apparent trends, it's too soon to declare victory. The proportion of high and medium severity vulnerabilities has changed little in a decade (Figure 1), and roughly half of the vulnerabilities are easy to exploit (Figure 2), suggesting that many developers are still ignoring security basics.

In a separate analysis, NIST looked at more than 3,000 NVD reports for denial-of-service vulnerabilities and found that 93.1 percent involved only a single condition, nearly always a too-long input string[4] (a few were exploitable only when two or three conditions held).

Clearly, software developers may be making real progress in securing systems, but broader adoption of secure programming practices—even simple measures such as input validation—could bring more dramatic improvements.

Although there's much room for progress, it appears that developers are taking security engineering seriously and meeting with success.

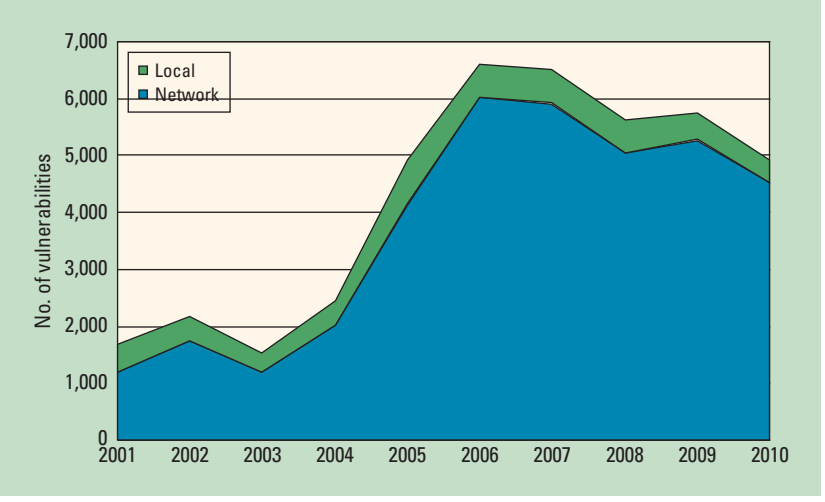

Figure 3. Vulnerabilities by access vector. The increase in vulnerabilities has been almost entirely in those exploitable by network access.

Acknowledgments

We're grateful to Harold Booth for consultations on the National Vulnerability Database data. We identify certain products here, but this doesn't imply recommendation by the US National Institute of Standards and Technology or other agencies of the US government, nor does it imply that the products identified are necessarily the best available.

References

1. R. Kuhn, H. Rossman, and S. Liu, "Introducing 'Insecure IT,'" *IT Professional*, Jan./Feb. 2009, pp. 24–26.
2. P. Mell, K. Scarfone, and S. Romansky, "A Complete Guide to the Common Vulnerability Scoring System Version 2.0," Forum of Incident Response and Security Teams, June 2007; www.first.org/cvss/cvss-guide.html.
3. "Web Server Survey," Netcraft, Apr. 2010; http://news.netcraft.com/archives/2010/04/15/april_2010_web_server_survey.html.
4. "Number of Interactions Involved in Software Failures—Empirical Data," Nat'l Inst. Standards and Technology, 2010; http://csrc.nist.gov/groups/SNS/acts/ftfi.html.

Rick Kuhn *is a computer scientist at the US National Institute of Standards and Technology. His research interests include information security, software assurance, and empirical studies of software failure. Kuhn has an MS in computer science from the University of Maryland College Park and an MBA from William & Mary, Mason School of Business. Contact him at kuhn@nist.gov.*

Chris Johnson *is a computer scientist at the US National Institute of Standards and Technology. His research interests include information security, software vulnerability analysis, and security automation. Johnson has a BS in computer science from the University of Maryland Baltimore County. Contact him at christopher.johnson@nist.gov.*