Handling and Reporting Security Advisories: A Scorecard Approach

Vendors and independent response centers have vastly different views regarding security advisories—what to publish and how to organize the information. The authors' scorecard approach aims to provide a practical guide for how to publish, read, evaluate, and handle advisories.

Dimitrios Lekkas
University of the Aegean

Diomidis Spinellis
Athens University of Economics and Business

A security advisory is a formal message issued by a vendor or a third party to alert a product's user community about security problems associated with the product and to provide information about how to avoid, minimize, or recover from any damage. This practice started emerging in the early 1990s when the spread of networking technologies exposed the exploitation of system vulnerabilities to the public.

A security advisory's life cycle starts from the *vulnerability disclosure*—the discovery of a security problem after users' reports or as a result of research and product evolution. The vendor determines a solution, builds the appropriate patches, and then publishes a detailed security advisory, which could simultaneously appear in other vendor-independent forums such as the Computer Emergency Response Team (CERT/CC; www.kb.cert.org/vuls) and the Common Vulnerability and Exposures (CVE; http://cve.mitre.org) dictionary. As vulnerabilities emerge, security advisory bulletins typically include descriptions of each, their potential impact on specific targets, and recommended solutions. Security response centers, which publish such bulletins, operate independently or as specialized departments within software and hardware vendors to help the community secure its systems and networks. Various revisions of an advisory might be published during its life cycle, as vendors release relevant patches and workarounds or, at a later stage, incorporate a solution into major product releases. A security advisory remains interesting to the community during the relevant vulnerability's life cycle until the number of systems it can exploit shrinks to insignificance.[1]

Security experts at the response centers usually judge and assign severity ratings to security advisories before they become public. However, a basic vulnerability rating, such as critical, moderate, or low risk, doesn't give users or system administrators enough information to assess the risk;[2] the vulnerability's impact will typically depend on specific product versions, system use profiles, configurations, hardware platforms, functional conditions, and local policies.

In fact, potential victims might even have difficulties identifying a vulnerability disclosure that poses a significant risk to them, whereas they might invest significant resources to defend against a vulnerability that doesn't represent a real threat. The increasing amount of information regarding vulnerabilities and exploits is further worsening the situation as it could confuse users and mislead them to take unnecessary or even risky defense actions. Attacks on systems rarely result from attackers' exploitation of previously unknown vulnerabilities. Rather, as with the Nimda worm and the recent Windows remote procedure call (RPC) buffer overrun, attacks typically exploit vulnerabilities for which solutions have long been available, but not applied. In fact, more than 90 percent of security exploits are caused by vulnerabilities for which there are known patches.[3]

To help users and system administrators efficiently manage and assess the impact of vulnerability disclosures, we have developed a *vulnerability scorecard*, which provides prescriptive guidelines based on a *goal-question-metric* approach. It is designed to let users record useful information and security response centers publish advisories in a way that will help the community respond more efficiently.

The (un)readability of security bulletins

Our quick survey of various security bulletin boards shows that each has a completely different view about what to publish, what information to include, and how to organize the data. With respect to the volume of published advisories, we recorded similar values at the various bulletin boards for specific vendors: an average of 45 for Cisco, 72 for Microsoft, and 44 for FreeBSD for each of the past three years. For general, non-vendor-specific informational postings, we recorded 37 advisories for CERT, 734 for Australian CERT (AusCert), 56 for Symantec, and 1,568 for CVE. The unexpectedly high difference between these numbers indicates that there's no clear rule on what's considered a security advisory—there's even confusion about what the terms "vulnerability disclosure" and "security advisory" mean.

As proof, let's look at a recent security advisory (the original can be found at www.freebsd.org/security/). Although written in a seemingly clear way, it misled us about the vulnerability's effect on our systems.

FreeBSD's security advisory SA-03:14.arp, published in September 2003, affects all FreeBSD system releases, independent of configuration, services running on the system, and installed software. The advisory notes that attackers could cause systems to hang by flooding them with Address Resolution Protocol (ARP) requests, a scenario deemed urgent. However, after reading through the advisory's three pages, its obvious that there's no real danger for FreeBSD systems on home PCs, because

- The advisory describes a temporary denial-of-service (DoS) attack and doesn't involve arbitrary code execution or data modification, which are the real concerns for personal systems.
- The attack can originate only from a local network, not from the Internet, which means it is no threat to systems that communicate only with the Internet—ours do so via dial-up lines and packet-filtering routers, for example.

Another example of how vendors' advisories are difficult to read is Microsoft's MS02-030 security bulletin, which applies to systems running SQL Server 2000. Microsoft rated this vulnerability moderate, which wouldn't indicate an urgent threat to system administrators. After carefully studying the advisory's seven pages, however, we concluded that our Web server, which we configured to accept ad hoc URL queries against a database frequently used by the academic community, was in real danger for two reasons:

- an option called SQLXML was enabled and could allow attackers to run arbitrary code by injecting scripts through XML tags, and
- the subject system provided critical Web-based applications to hundreds of interested parties.

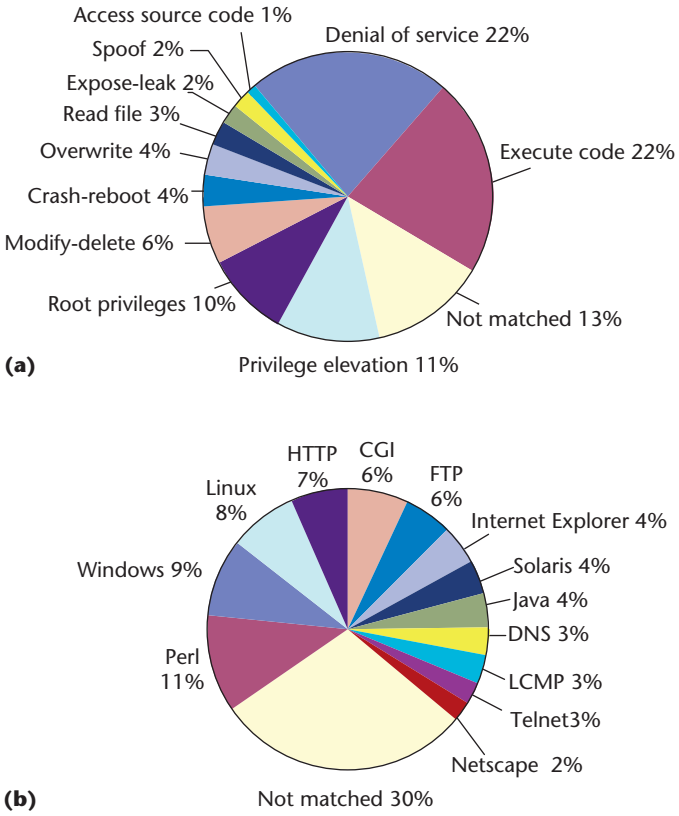

Figure 1. CVE keyword classification. We can match most advisories with (a) a specific impact and (b) applicability category by examining their title. Users can reach initial conclusions but not a confident assessment.

Clearly, users can't easily determine how applicable and severe security advisories are by superficially studying them. Users can reach initial conclusions about the impact and some applicability factors by detecting specific keywords in each advisory's summary. As Figure 1 illustrates, we performed a rough keyword classification in the collection of approximately 2,600 advisories in the CVE dictionary. It's interesting that by just examining the vulnerabilities' titles, we can classify 87 percent of them in 11 impact categories and 70 percent in 13 target platforms or applications. However, this classification ignores other important factors, including the impact on the community that uses the system, the exploitation preconditions, and the solution requirements and impact.

A metrics-based scorecard

Missing from vendors' bulletin boards, and even from

those run by organizations that solely handle emergency incidents, is a practical guide for how to read, evaluate, and handle a security advisory. Advisories are addressed to a heterogeneous community, including system or network administrators and users who operate publicly used servers, critical infrastructures, PCs, network devices, and other purpose-specific systems. A vendor's attempts to conclude on a security advisory's applicability and severity can mislead an important portion of the community.

Our scorecard provides a practical solution to this problem by defining a series of metrics. An interested party can determine the risk a specific system faces by evaluating the scorecard for a given security advisory. The comprehensive scorecard (shown in Table 1, pp. 36–37) contains nine major categories of metrics, ordered by their evaluation sequence, and gives a complete picture of both the vulnerability and relevant risk. By answering the metrics in order, we can many times arrive at a conclusion before traversing the whole list.

The scorecard's categories are mutually exclusive and cover a wide spectrum of security-related attributes. We chose the categories in the scorecard by recording and organizing the contents of security advisories published by various vendors and using other literature as a guide. Items 1 through 6 are to help assess the advisory's applicability and impact, whereas items 7 through 9 refer to implementing the advisory's proposed solution. More specifically, we follow Stefanos Gritzalis's findings that a *vulnerability's target* can be classified in terms of both logical and physical items.[4] Following the general target description, we continue with more specific information on the advisory's *applicability* and its *exploitation preconditions*, both of which appear in almost every security advisory. We must also consider some *organizational aspects*, which although they are not advisory-specific, significantly affect the assessment process.

We've refined the items in the *exploitation impact* category from relevant references that distinguish among vulnerabilities' impacts on hardware and software or classify them either according to their effects on availability, integrity, and confidentiality or, more generically, in terms of misuse, exposure, and denial of service (DoS).[5,6] We based the *community impact* metrics on Sokratis Katsikas's risk management analysis,[7] which identifies legal, financial, and trust aspects. Finally, the *solution requirements* are based, in part, on relevant taxonomies.[8,9]

Determining the value to assign to each metric depends on the target system's type and usage—whether it's a server, a router or switch, a shared terminal, an online PC, an offline PC, or other networked devices. For example, a DoS attack will affect a mail server with hundreds of users differently than it would affect a PC, even though both machines might have the same operating system, hardware, and configuration.

With that in mind, we could use a collection of quadratic vectors in the form `{advisory, system usage, metric, value}` to examine certain security advisories from the perspectives of various systems. Because we work with a single advisory or vulnerability and a single system at a time, however, we can simplify this approach by removing the first two dimensions (advisory and system usage). Assigning values to all or part of these metrics thus generates a single scorecard for a given vulnerability and system.

Using the results of a prior risk analysis study[10] adds value to the assessment process by determining the system's physical and logical limits, and recording and evaluating the system objects that constitute some value (assets).

Another practical assumption is that we can assign *discrete values* to all metrics, such as Boolean (yes/no) or low/high escalations, which provide a more readable overall result that leads to quick conclusions. Examples include

- countermeasure cost (high, moderate, low, or none);
- exploitation impact (high, moderate, low, or none);
- applicability (all, some, one, or none);
- time of action (immediate, short-term, long-term, or N/A); and
- loss of data (fully, partially, or none).

The *evaluation sequence* of the presented metrics for the assessment is also important. As a general guide, investigators should follow the following sequence, in which the […] notation indicates an optional step:

[Event detection →] Security advisory retrieval → Target → Applicability → Preconditions → Organizational factors [→ Exploitation impact → Community impact [→ Solution requirements → Solution impact [→ Solution implementation → Conclusions]]] .

The following contains the conditions under which the distinct steps of the evaluation process are executed:

1. The sequence begins either when a human or a process detects a real event or receives notification of a vulnerability disclosure and its relevant security advisory publication.
2. The advisory handler determines the target, applicability, and preconditions, before examining other organizational factors.
3. If applicable, the advisory handler evaluates the exploitation's impact on the system and the community.
4. If the collected impact metrics indicate that a solution is needed, the advisory handler will conse-

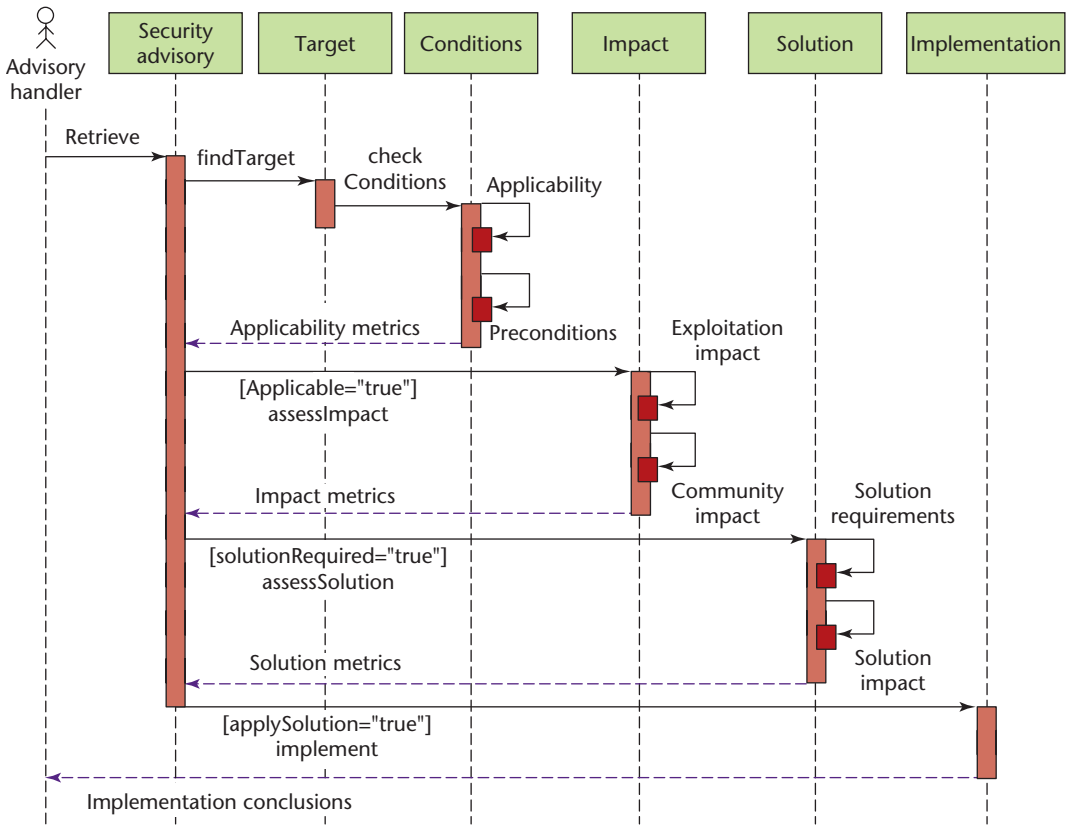

Figure 2. Action sequence for handling security advisories. The advisory handler performs a well-defined series of obligatory and conditional actions toward assessing the advisory and the possible implementation of the relevant solution.

quently evaluate the proposed solution's requirements and possible impact (steps 7 through 9 in the scorecard).

5. If the solution-assessment results indicate that the solution must and can be applied, the advisory handler implements it. At this point, the investigator must counterbalance the exploitation-impact metric values against the recorded impact and solution requirements to draw a clear conclusion on whether to implement the solution. Low exploitation-impact ratings and high solution-impact ratings would obviously argue against implementing the countermeasures.

Figure 2 illustrates the full procedure in a Unified Modeling Language (UML) sequence diagram. The diagram gives a precise picture of the required steps and conditions toward assessing an advisory and implementing the relevant solution.

The goal-question-metric approach

Administrators should view system security not only from the perspective of static characteristics but also from the perspective of emerging threats. Each new vulnerability or exposure could pose different objectives for security assessment and improvement. Various techniques enable administrators to implement quantitative and qualitative analysis of the system's security against a specific threat. The goal-question-metric (GQM) technique, a common analysis tool in software engineering and quality management,[11] and the Balanced Scorecard[12] — a multidimensional framework for describing, implementing and managing strategy at all levels of an organization—are good tools for supporting process improvements. Such quantitative approaches can be used either by response centers as a guide for publishing advisories or by users to efficiently handle security advisories against specific systems.

We based our scorecard's design on the GQM technique. The GQM user sets an objective goal that can't be directly interpreted, but rather is described by a series of questions. Each question is answered, in turn, by a series of metrics, which are either quantitative (obtain absolute values) or qualitative (answered by subjective judgments or comparable values).

A goal contains four parts:

Table 1. Scorecard metrics.

ASSESSMENT PHASE METRICS	DESCRIPTION
1. TARGET	
Logical:	Logical targets refer to informational and processing resources. Physical targets could refer to hardware, the LAN infrastructure, or, in an extreme case, to the entire Internet infrastructure.
Account	
Process	
Data	
Physical:	
System infrastructure	
Network (local range)	
Internet (wide range)	
2. APPLICABILITY – SCOPE	
Hardware architecture and platform	The applicability of a security advisory depends on hardware type, the operating system, what software is installed on the system, and various configuration settings. It is usually clearly indicated in the advisory's text.
Version of installed firmware	
Operating system and version	
Installed software	
Enabled features	
Configuration parameters	
Peripherals and hardware-specific software	
3. EXPLOITATION PRECONDITIONS	
Internet remotely exploitable	Vulnerabilities are usually exploited remotely, either independent of location, or only within specific logical or physical limits such as an Intranet area, a LAN, or a switched LAN segment. In other cases, an exploitation could succeed only if performed by normally registered users or through physical access.
Intranet remotely exploitable	
LAN shared medium exploitable	
LAN switched medium exploitable	
Registered user exploitable	
Requires physical access	
4. ORGANIZATIONAL FACTORS	
Comprehensiveness and completeness of advisory description	These factors could considerably mitigate a vulnerability's impact by providing better information dissemination and response procedures.
Existence of an incident response team	
Existence of prior risk analysis	
5. EXPLOITATION IMPACT (DAMAGE)	
Availability disruption (denial of service)	The first five items—availability disruption through stolen credentials—refer to basic security properties; that is, the availability, integrity, and confidentiality of the information and the infrastructure. Exploitation could also result in unauthorized action and system misuse, such as code execution and bypassing authentication and authorization controls. In other cases, the exploitation could provoke spreading to neighbor systems, erroneous transmission (network disruption, traffic redirection, and transmission out-of-sequence), or physical damage.
System or data integrity violation and loss of data	
Data disclosure and confidentiality breach	
Privilege elevation	
Stolen credentials	
Code/script execution	
Bypass of intended controls	
Misuse of resources	
Violation of system's security policy	
Affecting neighbor systems (spreading)	
Erroneous transmission	
Physical damage	

cont. on next page

Table 1. Scorecard metrics (cont'd).

ASSESSMENT PHASE METRICS	DESCRIPTION
6. COMMUNITY IMPACT	
Financial loss (labor time loss)	Financial loss might come in the form of direct theft and both downtime and restoration cost. Loss of trust in the information system is also a severe impact. The remaining items refer to illegal or criminal action and in more extreme cases, to national and international aspects.
Loss of trust	
Personal abuse, defamation, and humiliation	
Unauthorized gain of political authority and status	
Blackmail and other criminal action	
Action against the law	
Effects on national security and defense	
Effects on international relations	
IMPLEMENTATION PHASE	
7. SOLUTION REQUIREMENTS	
No action	The solution requirements focus on implementing the solutions, such as patching and configuring, according to the relevant security advisories. Additional protection measures might be required, such as using Access Control Lists (ACLs), intrusion detection systems, firewalls, cryptography, virtual private networks, and antivirus applications. The collection of reliable evidence data, by means of system logging, might also be part of the solution.
Workaround	
Locate relevant security advisories by other sources	
Patch availability and installation	
Operating system or application upgrade	
Software development	
Command execution	
Configuration modification	
Rebuilding of kernel or other executables	
Configuration or installation of additional protection measures	
Enable logging of evidence data	
8. SOLUTION IMPACT	
Cost in time and money for establishing countermeasures	Implementing a proposed solution is not free—the cost in terms of money, labor, system availability, and organization functionality is usually significant. The deadline is also an issue: depending on the severity of the impact, it would be immediate or long term.
Cost in time and money for regression testing the updated system.	
Availability of the system while applying the solution	
Consequences on the organization's functionality	
The deadline to take action	
9. CONCLUSIONS IMPACT	
Security advisory's severity	The conclusions that will arise after the assessment and the implementation phases of a security advisory are either informational or indicate further action is needed.
Countermeasures' efficiency	
Need for future plan to protect the system	
Need for further communication with the vendor	
An indication that attacks will be repeated or increased	

Table 2. Goal-question-metric (GQM) analysis of an unchecked buffer in SQLXML that could lead to code execution.

	METRIC	VALUE
Question A (general factors): Is the vulnerability disclosure well described and documented?		
	Metric A1: Comprehensiveness	Good
	Metric A2: Completeness	High
Question B (applicability and preconditions): Is my system in danger?		
	Metric B1: Is Microsoft SQL server 2000 installed?	Yes
	Metric B2: Internet connectivity	Partial
	Metric B3: Are XML queries through HTTP enabled?	Yes
	Metric B4: Is the vulnerability exploitable by privileged users only?	Yes
Question C (target): Which system objects are in danger?		
	Metric C1: Logical targets	Data
	Metric C2: Physical targets	None
Question D (exploitation impact): Is the risk high?		
	Metric D1: Code execution	High
	Metric D2: Data modification and loss	High
	Metric D4: Credentials stolen	No
Question E (solution impact): Is immediate action necessary or can I leave it for the next working day?		
	Metric E1: Vendor's severity rating	Moderate
	Metric E2: Number of people using the server	< 10
Question F: (conclusions impact): Is protection against the vulnerability really needed?		
	Metric F1: Subjective reply based on metrics A to E	Not urgent

- An *issue* relates to a security parameter (such as the impact);
- A *reference object* is the source of the analysis, such as vulnerability bulletin CVE-2001-0852;
- A *perspective* establishes how to interpret the issue—in terms of its impact on a service, process, system, or the like; and
- An *intention* determines how to evaluate or change the object's parameter (assess, test).

The GQM user must construct a series of questions and relevant metrics according to the given system's characteristics and requirements (the perspective criteria in the goal). For example, a question about a DoS attack's impact would probably appear when referring to a core mail server, but not in the case of a personal system used at home. To illustrate, let's examine two security advisories under the perspectives of two different system types.

Example 1: Assess

In our first example, the goal is to assess (determine intention) the impact (issue) of the vulnerability (object) described by the CVE advisory CAN-2002-0187 ("Unchecked buffer in SQLXML could lead to code execution" www.microsoft.com/technet/security/bulletin/MS02-030.asp) against a personal Windows Web server (perspective).

Example 2: Protect

In our second example, the goal is to protect (intention) the border router (perspective) of the national tax information system from the vulnerability (issue) described in the 16 July 2003 Cisco security advisory (object) ID 44020 ("Cisco IOS interface blocked by IPv4 packets").

Ideally, a GQM list, such as that shown in Tables 2 and 3, should accompany a vendor's security advisory. The list should contain the most relevant queries to the advisory metrics and estimated values, helping advisory handlers quickly identify the required data and determine the risk faced. In any situation, the vendor should present at least two separate GQM lists, representing the two extreme perspectives—whether an affected system is a personal system or a critical widely used infrastructure.

Case study: Testing the repeatability of scoring reports

A key issue that adds value to our proposed solution is the scoring reports' *repeatability*. Our scorecard's rating is characterized as repeatable when different observers can rely on it without undertaking a detailed examination of the security advisory. An efficient exploitation of our proposal, which produces highly repeatable scoring reports, will reduce both the complexity of judging risk and the work factor necessary for examin-

Table 3. GQM analysis of Cisco IOS interface blocked by IPv4 packets.

	METRIC	VALUE
Question A (target): What is the target of the threat?		
	Metric A1: What parts of the system infrastructure are affected?	All border router interfaces
Question B (applicability): Is the current router configuration tolerant against the threat?		
	Metric B1: Current version of IOS	Vulnerable
	Metric B2: Is the protocol independent multicast (PIM) enabled?	Yes, higher risk
	Metric B3: Could an Ethernet interface affect neighbor systems through ARP requests?	Yes
Question C (preconditions): Are the additional preconditions for exploiting the vulnerability satisfied?		
	Metric C1: Is IPv4 enabled?	Yes
	Metric C2: No access control list (ACL) blocks on TCP protocols 53, 55, 77, and 103	Yes
Question D (exploitation impact): What is the maximum exploitation impact?		
	Metric D1: Denial of service	High
	Metric D2: Affecting neighbor systems	Yes
	Metric D3: Code execution	No
Question E (community impact): What are the consequences of a possible attack on citizens?		
	Metric E1: Financial loss	Likely
	Metric E2: Loss of trust	High
	Metric E3: Criminal action	No
Question F (solution requirements): What is the immediate action needed?		
	Metric F1: IOS upgrade	Yes
	Metric F2: ACL update	Yes
Question G (solution impact): What are the effects of the solution implementation?		
	Metric G1: Cost in money	Zero
	Metric G2: Downtime	10 minutes
	Metric G3: Impact to other systems	None
	Metric G4: Consequences to the functionality of public services, if applied during low-traffic time	Low
	Metric G5: Rate (cost of impact/cost of solution)	High

ing a vulnerability disclosure and its related security advisories. High repeatability of scoring reports can also contribute to intrusion detection procedures' automation and scalability.[13]

The experiment

To test our approach, we used historical data for vulnerability disclosures from the CVE dictionary to plug into our metrics-based scorecard. Our objective was to collect results regarding how much effort was required to quantify the scorecard metrics, as a result of the repeatability of the reports. The higher the repeatability rate under different perspectives, the more straightforward quantification of a new scorecard. We examined 200 CVE entries spanning a five-month period in 2002 and added some extra entries applicable to NetBSD systems. To collect the data to perform the case study, we extracted vulnerability descriptions from the CVE dictionary and located the related security advisories from product vendors. We studied each advisory up to the point at which we could draw a clear conclusion regarding its applicability, recording the metrics in our scorecard (detailed scoring of the examined advisories is available at www.syros.aegean.gr/users/lekkas/cve200_scoring.htm).

We repeated this procedure for three different systems:

- an Intel-based system running Microsoft Windows 2000, serving as a database and Internet Web server for a medium-sized university,
- a Cisco router (7500 series) serving as a medium-sized university's border router, and
- an ARM-based NetBSD network appliance periodically connected to the Internet.

We initially found 35 applicable entries for the Windows system, and later discovered that 18 of them were inapplicable because they referred to specific configuration parameters. In seven entries, we stopped the assessments at an intermediate step, primarily because their impact proved to be unimportant (for example, a browser vulnerability affects workstations but not servers). We com-

Table 4. Repeatability of scoring reports under multiple observations.

OBSERVATION PERSPECTIVE	DATABASE & WEB SERVER	MAIL SERVER	INTRANET FILE SERVER	WORKSTATION LEKKAS	WORKSTATION SPINELLIS
Database & Web server		97% (34)	94% (33)	57% (20)	57% (20)
Mail server	97% (34)		91% (32)	54% (19)	54% (19)
Intranet file server	94% (33)	91% (32)		57% (20)	57% (20)
Workstation Lekkas	57% (20)	54% (19)	57% (20)		94% (33)
Workstation Spinellis	57% (20)	54% (19)	57% (20)	94% (33)	

pleted the assessment and resolution-implementation steps for 10 of the other Windows entries. We had to perform all the assessment and implementation steps for approximately 25 percent of the vulnerabilities applicable to the Cisco and NetBSD systems.

Our experiments focused on our basic research question: the repeatability of a scoring report under multiple observers. To examine whether a different observer could rely on an existing precompiled scoring report rather than the complete security advisory, we repeated the scoring procedure for the Windows 2000 system (against the 35 applicable vulnerabilities) under five perspectives:

- a database and Internet Web server running SQL Server 2000 and IIS 5.0,
- an Internet mail server running Microsoft Exchange,
- an Intranet Windows NT-based file server,
- a PC as observed by one author of this article (Lekkas), and
- a PC as observed by this article's second author (Spinellis).

As Table 4 shows, a high percentage of the scoring reports proved to be repeatable, assuming that the examined systems have similar applicability factors in terms of platforms and basic configurations. Each value in Table 4 (both the percentage and absolute number) represents the number of repeatable scoring reports between two different observing perspectives. The observations' order doesn't affect the result; therefore, the resulting matrix is symmetric ($a_{ij} = a_{ji}$). The empty cells in the table correspond to comparisons between the same observations.

Results

Our results show that the scoring reports have total repeatability when observed under the perspective of Internet servers, high repeatability between Internet and Intranet servers, partial repeatability between servers and workstations, and high repeatability between different workstations. After further analysis, we've concluded that differences in the scorings recorded between Internet and Intranet servers were caused by different exploitation preconditions—specifically, the dependency on whether the vulnerability is exploitable remotely or locally (and usually by authenticated users). On the other hand, most of the differences between servers and workstations derived from requiring local user intervention for a vulnerability's exploitation (such as opening a document or interacting with a remote site)—an occurrence that's common on a workstation but improbable on a server. The number of nonrepeatable reports between the two workstations was very small and was caused by applicability differences, which were usually obvious by simply reading the vulnerability's title.

We can now argue that targeted preprocessing of our scorecard under three main perspectives results in highly repeatable reports and could substantially increase the efficiency of relevant security advisories. It's also possible to identify different categorizations of system usage, depending on the device type. For example, a relevant categorization for a network device would be border router, access server, and LAN equipment. Figure 3 shows a simplified precompiled scoring report for a specific vulnerability.

Our study's results give us a tangible way to improve issuing and handling security advisories. Although current systems technology cannot eliminate the role of a human system administrator in the loop who processes incoming security advisories and patches, advisories can be improved. Our vision for the proposed framework is that vendors and response centers will use and evolve it toward a homogenized and stable security advisory publication scheme, using, for example, a common XML format. A user or administrator can easily filter incoming precompiled scorecards based on the characteristics of its deployed systems to eliminate numerous irrelevant messages. More important, administrators will immediately grasp a vulnerability's impact on their systems, without dedicating too much thought to the underlying details of the advisory. Thus, security advisories will improve, both by reducing the complexity of judging risk and by helping humans easily identify, prioritize, and concentrate on the really important issues. □

CVE-2002-0650 description:	The keep-alive mechanism for Microsoft SQL Server 2000 allows remote attackers to cause a denial of service (bandwidth consumption) via a "ping" style packet to the Resolution Service (UDP port 1434) with a spoofed IP address of another SQL Server system, which causes the two servers to exchange packets in an infinite loop.		
PERSPECTIVE	**INTERNET SERVER**	**INTRANET SERVER**	**WORKSTATION**
Target	System, network, Internet	System, network	System, network
Applicability	SQL server installed and enabled	SQL server installed and enabled	SQL server installed and enabled
Preconditions	Remotely exploitable	Spreading by neighbor systems	Spreading by neighbor systems
Organizational	Advisory existed long before current massive exploits	Advisory existed long before current massive exploits	Automatic system update does not download the patch
Damage	DoS (system and network disruption)	System disruption and low risk of LAN disruption	Low impact. No data disclosure or code execution
Community impact	High (financial loss)	Important	None
Solution requirements	Patch installation or ACL blocking port 1434	Patch installation	Patch installation
Solution impact	Server needs restarting; remote connections disabled if ACL enabled	Server needs restarting	None
Conclusions	Critical situation; need further observation	Important risk	Not a critical situation

Figure 3. A simplified precompiled scorecard for basic system usage perspectives.

References

1. W. Arbaugh, W. Fithen, and J. McHugh, "Windows of Vulnerability: A Case Study Analysis," *Computer*, vol. 33, no. 12, 2000, pp. 52–59.
2. S. Mash, "Risk Assessment for Dummies," *Computer Fraud & Security*, vol. 2002, no.12, 2002, pp. 11–13.
3. L. McGhie, "Software Patch Management—The New Frontier," *Secure Business Quarterly*, vol. 3, no. 2, 2003.
4. S. Gritzalis, "Information Systems Security in Distributed Environments," doctoral dissertation, Dept. of Informatics, Nat'l and Kapodistrian Univ. of Athens, 1998.
5. U. Lindqvist and E. Jonsson, "How to Systematically Classify Computer Security Intrusions," *Proc. 1997 IEEE Symp. Security & Privacy*, IEEE CS Press, 1997, pp. 154–163.
6. J. Howard and T. Longstaff, *A Common Language for Computer Security Incidents*, report no. SAND98-8667, Sandia Int'l Labs, 1998.
7. S. Katsikas, "Risk Management of Information Systems," *Information Security: Technical, Legal, and Social Issues*, E. Kiountouzis, ed., EPY Athens, 1995.
8. H. Venter and J. Eloff, "A Taxonomy for Information Security Technologies," *Computers & Security,* vol. 22, no. 4, 2003, pp. 299–307.
9. M. Laakso, "Introducing Constructive Vulnerability Disclosures," *Proc. 13th FIRST Conf. Computer Security Incident Handling & Response*, 2001.
10. R. Baskerville, "Risk Analysis: An Interpretative Feasibility Tool in Justifying Information Systems Security," *European J. Information Systems*, vol. 1, no. 2, 1991, pp. 121–130.
11. V.R. Basili, G. Caldiera, and D. Rombach, "The Goal Question Metric Approach," *Encyclopedia of Software Engineering*, vol. 2, John Wiley & Sons, 1994, pp. 528–532.
12. L. Buglione and A. Abran, "Balanced Scorecards and GQM: What Are the Differences?" *Proc. 3rd European FESMA-AEMES Software Measurement Conf.*, 2000.
13. M. Bishop, "Trends in Academic Research: Vulnerabilities Analysis and Intrusion Detection," *Computers & Security*, vol. 21, no.7, 2002, pp. 609–612.

Dimitrios Lekkas *is a lecturer at the University of the Aegean's Department of Product and Systems Design Engineering. His research interests include computer security, incident response, public-key cryptography, database management systems, and distributed systems. Lekkas has a PhD in information systems security from the University of the Aegean and an MSc in information technology from Glasgow University. He is a member of the Greek National Educational Network (EDUnet) technical committee and coordinator of the Greek academic network public-key infrastructure (GRnet-PKI). Contact him at dlek@aegean.gr.*

Diomidis Spinellis *is an associate professor at the Department of Management Science and Technology at the Athens University of Economics and Business, Greece. His research interests include software engineering tools, programming languages, and computer security. Spinellis has an MEng in software engineering and a PhD in computer science, both from Imperial College, London. His book,* Code Reading: The Open Source Perspective *(Addison-Wesley 2003), received a Software Development Productivity Award from* Software Development *magazine in 2004. Spinellis is a member of the IEEE, the IEEE Computer Society, and the ACM. Contact him at dds@aueb.gr.*