Towards a Common Enumeration of Vulnerabilities

David E. Mann, Steven M. Christey
The MITRE Corporation
202 Burlington Rd., Bedford MA 01730
{damann, coley}@mitre.org

January 8, 1999

Abstract
In this paper, we discuss the use of multiple vulnerability databases in our operational enterprise security environment and we consider some of the roadblocks we see to achieving interoperability between them. We introduce the concept of a Common Vulnerability Enumeration (CVE) as a mechanism that we believe will help to foster easier data sharing. We consider some historical examples of the development of taxonomies in other fields and relate them to current efforts in representing and sharing vulnerability information. We present a simplified representation of a "vulnerability" and discuss how we anticipate using it to mitigate the problem of interoperability. We also describe some of the practical issues that may be involved in the development and use of a CVE.

1 Introduction

The MITRE Corporation is an independent, not-for profit company working in the public interest that provides technical support to the government. The work discussed in this paper has been done for MITRE's own internal Information Security Committee. MITRE uses a variety of tools to assess its security posture, including several network assessment tools and Intrusion Detection Systems (IDS) from a wide variety of vendors. These tools all represent and process vulnerabilities in one form or another. We are building a system that can integrate and manage vulnerability information from different sources in a centralized database that will then be linked to our internal enterprise databases for the purpose of supporting our enterprise security operations. For example, we need to achieve the following capabilities immediately:

- When we receive vulnerability information from multiple assessment and/or IDS tools, we need to know if they have identified the same vulnerability or not.
- If we receive an IDS alarm, we need to determine the status of that host in terms of recent assessment scans with regards to the related vulnerability.

The authors wish to acknowledge their colleague, William H. Hill, who provided much of the conceptual inspiration for this paper.

- If a network assessment reveals that a host is open to a particular vulnerability, we need to gather and distribute all of the information we have on that vulnerability.
- We need to be able to compare tools in terms of their completeness, e.g. which one can identify the most vulnerabilities, or how much coverage there is with respect to vulnerabilities identified in CERT advisories.

Every information security tool we are attempting to integrate has its own implicit or explicit vulnerability database. The most pressing problem we face in attempting to gain this integration is that there is no common naming convention and no common enumeration of the vulnerabilities in these disparate databases. As a result, it is labor intensive to compare output from assessment tools (or IDS tools) from different vendors, to relate IDS alarms with network assessment probes, and to relate any tool with other information sources.

There are several efforts that attempt to define shareable schemas, universal primitives and robust taxonomies for vulnerability information. While we remain interested in and supportive of these efforts, we note that there appears to be no consensus on these issues yet. Our problem is that we need to achieve interoperability among our tools right now. Moreover, the problem preventing us from achieving this integration is much more basic than the definition of good schemas, primitives or taxonomies. The problem is that there is no consistency in the community with regards to identifying the vulnerabilities.

In section 2 of this paper, we discuss some of the roadblocks we see to achieving interoperability. In section 3, we introduce the idea of a Common Vulnerability Enumeration (CVE). In section 4 we consider some historical examples of the development of taxonomies. In section 5, we consider a simplified representation of vulnerability information and discuss how we anticipate using a simplified representation to address the problem of interoperability. Section 6 of this paper describes some of the practical issues that may be involved in the development and use of a CVE. We also discuss a shortcoming of our approach. In section 7, we discuss areas for future work. And in section 8, we summarize our conclusions.

2 Roadblocks to Interoperability

We outline four separate challenges that we face in trying to achieve interoperability between all our tools and other vulnerability information sources.

2.1 Inconsistent Naming Conventions

Consider the problem of naming vulnerabilities in a consistent fashion. For example, one vulnerability discovered in 1991 allowed unauthorized access to NFS file systems via guessable file handles. In the ISS X-Force Database, this vulnerability is labeled *nfs-guess* [8]; in CyberCop Scanner 2.4, it is called *NFS file handle guessing check* [10]; and the same vulnerability is identified (along with other vulnerabilities) in CERT Advisory CA-91.21, which is titled *SunOS NFS Jumbo and fsirand Patches* [3]. In order to ensure that the same vulnerability is being referenced in each of these sources, we have to rely on our own expertise and manually correlate them by reading descriptive text, which can be vague and/or voluminous.

2.2 Managing Similar Information from Diverse Sources

Among other tasks, we have attempted to relate our scanner probes with CERT Advisories and other sources of vulnerability information to provide us with a comprehensive picture of a particular vulnerability. The internal documentation for CyberCop Scanner 2.4 indicates 13 separate probes that relate to NFS problems [10]. Each of these 13 probes lists the same 6 CERT Advisories as source documents for related information. As of this paper's publication, the ISS X-Force database generated 20 hits related to NFS [8], while a similar search of the INFILSEC database produced only 4 hits [6]. The problem of relating these vulnerabilities quickly becomes apparent: are there 13, 6, 20, 4, or some combination thereof? How many NFS vulnerabilities really exist, and how complete are each of these databases with respect to the "true" number of vulnerabilities?

2.3 Managing Multiple Evolving Perspectives of the Same Vulnerability

Each source of information related to vulnerabilities imposes a different perspective. For example, an IDS perspective of a vulnerability may be related to an attack signature; a scanner perspective may be closely related to the probe that is used to identify the same vulnerability; a taxonomy may classify it according to the type of fault that causes the vulnerability; and the enterprise perspective may define the risk according to its network configuration and security policy. These disparate perspectives make it difficult to design schema which fully capture all this information. To further complicate matters, each of these perspectives can change over time, requiring modifications to any representation that attempts to unify these disparate concepts. We need an approach that insulates us as much as possible from these representational problems.

2.4 Complexity of mapping between databases

Each database (or tool) containing vulnerability information may have a different enumeration of vulnerabilities. Therefore, in order to achieve full interoperability among all of the databases, we would need to determine and maintain a mapping between the entries for every pair of databases. Without a naming convention or other standard means of referencing vulnerabilities, the mapping process is labor intensive and error prone.

3 Common Vulnerability Enumeration (CVE)

Without agreement on how to list and name the vulnerabilities, our integration task is made much more difficult due to the number of mappings we need to perform. In order to achieve interoperability of security tools and share vulnerability information, we need a ***Common Vulnerability Enumeration (CVE),*** a standardized list that:

- enumerates and discriminates between all known vulnerabilities
- assigns a standard, unique name to each vulnerability
- exists independently of the multiple perspectives of what a vulnerability is
- is publicly "open" and shareable without distribution restrictions.

A Common Vulnerability Enumeration would allow us to evaluate the comprehensiveness of our various information sources. A standard name would help us to more automatically and clearly relate vulnerability information from those sources.

Independence from multiple vulnerability perspectives will help a CVE to remain stable. And finally, a CVE must be public, without distribution restrictions, to allow free data exchange across the community.

3.1 The Lack of a Common Vulnerability Enumeration

In our opinion, currently there is no single enumeration of vulnerabilities that has all the characteristics of a good CVE.

Many commercial IDS and scanning tools have implicit or explicit databases that are related to vulnerabilities, but none of them are truly "vulnerability databases" which enumerate a complete list of vulnerabilities. The databases we have seen associated with network scanners might be more properly called "probe databases" since, to a large extent, the entries correspond to the various probes of the assessment tools. IDS tool databases might be better referred to as "exploit signature databases" since their focus is primarily on exploit signatures or IDS alarms. A database that MITRE is currently developing for internal use might be called an "enterprise database" since it seeks to relate enterprise-specific information to vulnerabilities, e.g. host and network configuration, IDS alarms, assessment results, and security policy. Each of these databases has a different perspective of what a vulnerability is.

Some databases may attempt to organize vulnerability information based on some central theoretical concept. For example, a vulnerability database may be organized around the notion that a vulnerability is a software fault [1]. But such a database will ultimately have a hard time dealing with vulnerabilities associated with misconfigurations or hardware faults. Yet another vulnerability database [9] defines vulnerabilities as a process by which unauthorized access can be gained and in so doing, it may be more correctly regarded as an exploit database. While many database efforts have attempted to enumerate a wide spectrum of vulnerabilities, they are generally incomplete with respect to the total number of known vulnerabilities.

Other information sources may be considered to form implicit vulnerability databases. For example, the body of CERT Advisories and Vendor-Initiated Bulletins composes an implicit database of vulnerabilities, as does any collection of advisories. And even "hacker" exploit repositories form de facto vulnerability databases, some of which are very comprehensive.

Many of the described databases are inherently incomplete because they are only designed to address a particular problem or class of problems. For example, network scanners and IDS tools are largely limited only to vulnerabilities that are exploitable via the network; CERT only publishes advisories for vulnerabilities that are high risk or widespread.

There are some efforts that attempt to enumerate all vulnerabilities. Some are not publicly accessible or have distribution restrictions. Others are relatively incomplete. Some of the most complete databases (e.g. the ISS X-Force database) are copyrighted and remain under control of the authors, and as such cannot be distributed freely [8].

Some efforts may indirectly address the problem of nomenclature, e.g. the Intrusion Detection Working Group of the IETF. However, as of this writing, the focus appears to be on representation and data exchange rather than integration. It is focused on data types and not the actual data that needs to be defined, and is understandably biased towards the IDS attack signature perspective with respect to vulnerabilities [7].

In short, at this time, none of the databases that we are aware of are satisfactory candidates as a Common Vulnerability Enumeration to use for our integration task.

4 Naming and Enumeration in Other Fields

As we consider the adoption of a common vulnerability enumeration (CVE), we may be able to glean some perspective by considering similar efforts of the past, such as the development of the periodic table of elements and the identification of animals.

4.1 Example 1 - The Elements

In "The Evolution of the Periodic System," Eric Scerri [11] recounts the development of the modern periodic table of elements invented by the Russian chemist, Dimitri Mendeleev. Scerri notes:

> The discovery of the periodic system for classifying the elements represents the culmination of a number of scientific developments, rather than a sudden brainstorm of the part of one individual… Prior to Mendeleev's discovery [in 1869], other scientists had been actively developing some kind of organizing system to describe the elements. In 1787, for example, French chemist Antoine Lavoisier, working with Antoine Fourcroy, Louis-Bernard Guyton de Morveau and Claude-Louis Berthollet, devised a list of the 33 elements known at the time. Yet such lists are simply one–dimensional representations. The power of the modern table lies in its two- or three- dimensional display of all the known elements (and even ones yet to be discovered) in a logical system of precisely ordered rows and columns.

While Scerri's article focuses on the organizational and analytical power of Mendeleev's periodic table, it is also clear that before such a table could be thought of, the elements first needed to be enumerated and named. Mendeleev's periodic table did not appear in print until 1869, more than 80 years after Lavoisier's simple and incomplete enumeration. It could be argued that the simple 1-dimensional listings of the elements that were commonly accepted were precisely the tools that allowed for effective communication in the scientific community, with the resulting interaction and information exchange providing the foundation for Mendeleev's later advances in categorization.

With respect to vulnerability information, it appears to us that the information security community is at a point that is similar to chemistry prior to the time of Mendeleev. The community may not agree on a classification scheme for vulnerabilities, but we probably know enough to begin to enumerate the vulnerabilities, independent of their classification. To borrow Scerri's language, all that is necessary at this point is a 1-

dimensional representation, at least from our operational perspective. The use of more complex representations can wait until some consensus is achieved. The inevitable redefinition of our first 1-dimensional enumeration is acceptable, so long as we can settle on some initial standard as a first step towards more immediate interoperability.

4.2 Example 2 - The Animals

Consider the case of the grizzly bear. We wish to make several observations. First, at one time, the California golden bear, the brown bear, the Kodiak bear and the grizzly bear were all considered to be different species. Today they are considered to be just one species (Ursus arctos) [4]. We note that the nomenclature for and categorization of zoological information is driven by the consensus of the zoological and biological communities. In contrast, there is no such consensus within the information security community with respect to vulnerability information.

Second, the process of naming (and categorizing) is an iterative process, one that is open to redefinition as more is learned. If the history of the naming of the species is any indication, we as a community must accept that our first attempts at naming the vulnerabilities may be flawed, and we should commit ourselves to the refinement of our nomenclature even as we endeavor to learn more about vulnerabilities. Furthermore, we must commit ourselves to restricting our renaming efforts within the context of community acceptance.

Finally, we note that much of the utility of our common language concerning the animal kingdom stems from our ability to identify animals with varying degrees of precision or abstraction. For example, is it more useful to use the term bear, grizzly bear, Kodiak bear or Ursus arctos? The answer clearly depends on the context. This final observation begs the following question. To what degree of abstraction should we define vulnerabilities?

4.3 Levels of Abstraction

Suppose we were to consider 3 different enumerations of all of the known animals, where the first is an enumeration of all known families, the second is an enumeration of all known species, and the third is an enumeration of all known sub-species. We might expect to find an entry for bear in the first, grizzly bear (species: Ursus arctos) in the second and Kodiak bear (a sub-species of Ursus arctos [4]) in the third. In contrast, if we were to consider 3 separate enumerations of all known vulnerabilities, we would have no knowledge of what level of abstraction to expect based solely on the word “vulnerability.” One enumeration may list vulnerabilities at a high level of abstraction (buffer overflow, etc.). The second may list faults in specific pieces of software as identified by name, version and even sub-version. And the third may fall somewhere in between. Similar ambiguities with regard to differing levels of abstraction are discussed in [2].

If a simple, 1-dimensional CVE will enforce a single level of abstraction, then at what level of abstraction should the CVE be defined? We believe that a de facto standard has already emerged in this regard. Based on our experience with the vulnerability databases of various security tools and our perusal of several publicly available vulnerability databases, we believe there is much commonality with respect to the level of abstraction that would be appropriate for a CVE. This level of abstraction is more specific than "buffer overflow" and more general than specifying individual faults in individual pieces of software. For example, most IDS and scanning tools have knowledge of vulnerabilities such as phf, NFS file handle guessing, and the wuftp "site exec" problem. Thus, we believe that a workable first attempt at establishing a CVE, along with its implicit decision with respect to the proper level of abstraction, can be achieved by basing a CVE on the existing security tools and publicly available vulnerability databases, as much as possible.

One may argue that a CVE based on the emerging standard level of abstraction will be unusable for some purposes. For instance, an automated patch management system may need to deal with security information at a much lower level of abstraction. We believe that this causes no problem with respect to a CVE (one that is presumably defined at a higher level of abstraction) unless we say that the patch management system contains an enumeration of "vulnerabilities." It is only at this point that we make the semantic mistake of using "vulnerability" at 2 different levels of abstraction.

4.4 Conclusions

Classification schemes and taxonomies strive to organize sets of information. If the development of our common language of the animal kingdom and the elements is any indication, the information security community is correct to continue to work towards a robust categorization scheme or taxonomy for vulnerability information. However, we suggest that the community would be mistaken to wait until we have a fully developed classification for vulnerabilities with a corresponding language that would allow us to handle vulnerability information at different levels of abstraction. While this is an interesting and important research problem, all that we need to meet our immediate and operational objectives is a simple, unstructured listing of the known vulnerabilities – a 1-dimensional Common Vulnerability Enumeration (CVE) of the known vulnerabilities. A CVE would move us closer to immediate interoperability between different databases and tools, and it would foster a greater degree of data exchange between tools and across the community by implicitly defining a common language of vulnerabilities.

5 Defining "Vulnerability"

As we consider the task of developing a sharable vulnerability database, we may be able to gain some insight by considering what a sharable species database might look like. Species can be defined as groups of interbreeding natural populations that are reproductively isolated from other such groups. [5] While this definition gives some guidance to zoologists with respect to categorization efforts (although asexual reproduction remains problematic) this definition is ambiguous with regards to identifying a list of universal primitives that would be capable of defining all species.

We argue that any definition of "vulnerability" that is broad enough to include both software faults and network configuration errors will be equally ambiguous in terms of determining a list of primitives capable of capturing all known vulnerabilities. Such a definition of "vulnerability" will affect a CVE in at least two ways. First, it should shape our understanding of how a vulnerability entity will relate to other entities that we may want to model. Secondly, it will have a large impact on the definition of the vulnerability entity itself.

5.1 CVE – A Logical Bridge

Our vision of a Common Vulnerability Enumeration is for it to provide a mechanism for linking together vulnerability-related databases or concepts – and nothing more. Rather than seeing this narrow scope as a limitation, we see this as an advantage. By agreeing to limit the use of the CVE to the role of a logical bridge, we are then free (or forced) to define our own ***extensions*** of the concept of a vulnerability according to our own specific operational needs. See Figure 1.

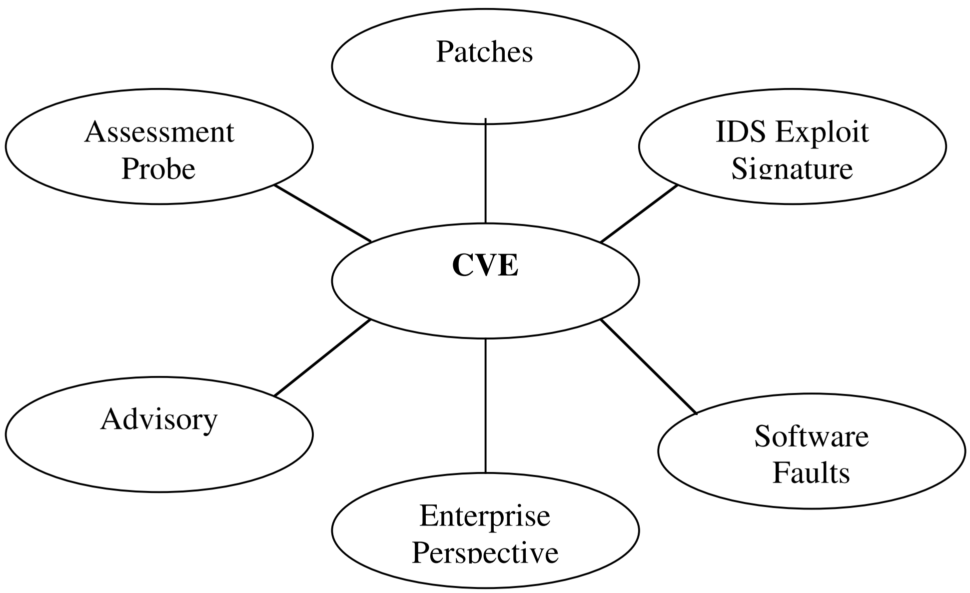


Figure 1: Extensions to a Common Vulnerability Enumeration

Consider an example from our current work. We have two separate databases: one that receives IDS alarms, and one that receives output from assessment tools. Both databases have entities in them that are titled "Vulnerability." Given the different operational objectives of the two databases, it is not surprising that the vulnerability entities in each database are comprised of very different lists of attributes. In fact, it would be more appropriate if the two competing Vulnerability entities were called Vulnerability_IDS_View and Vulnerability_Assessment_View instead. Moreover, since our IDS and assessment databases are both works in progress, we expect that the definitions of the Vulnerability_IDS_View and Vulnerability_Assessment_View entities will change over time. By maintaining separate logical extensions to the common, shared entity of Vulnerability, we compartmentalize our data at a conceptual level. This allows us to continue with our development of both databases independently while still preserving interoperability by maintaining links back to our CVE.

5.2 A Minimal Representation of Vulnerability for a CVE

As described in the requirements for a CVE in Section 3, a CVE should exist independently of the multiple perspectives of vulnerabilities. One way to ensure the independence of a CVE is to define vulnerabilities with only the necessary and sufficient attributes that are common to all vulnerabilities, ensuring that these attributes do not rely on any evolving representation and can be commonly agreed to by the majority of the security community.

We now present another example from our recent work. One of the objectives was to tie vulnerability information to information about individual hosts. In particular, we have been interested in knowing what operating system a vulnerability applies to since we want to tie this to our enterprise knowledge about which hosts are running which operating systems. A problem arose with regards to the granularity of the operating system information we wanted to track. At one extreme, we could have chosen to enumerate every possible operating system including every possible version and every possible combination of service packs and patches. At the other extreme, we could limit the granularity to a choice of, say, four platforms: Unix, Microsoft, Macintosh and Other. While the former approach has an obvious appeal in its completeness, we found that it was unworkable in terms of its practicality. On the one side, we could not find definitive information about the applicability of different vulnerabilities to *all* platforms and we concluded that the amount of testing and evaluation required to do so would be overwhelming. On the other side, it is difficult to conduct an automated software inventory of all our different hosts. Given the highly dynamic nature of host configurations and our relative inability to know what *exact* operating system is running on which host, we have given up, for the time being, in our attempt to handle operating system information at a precise level.

Our point in citing this example is that we doubt that we could find agreement at a community level on how to model something as innocuous as "affected operating system." Each of us will be driven by our own different concerns when it comes to vulnerabilities, and these differing concerns will lead us to differences in schemas and primitives. We believe we will be much better off by initially agreeing to agree on a minimal set of attributes for the concept of "vulnerability" itself.

We argue that a vulnerability in a CVE does not need any attributes beyond a unique name and a textual description. As long as the name is unique, and the description includes enough information to distinguish a vulnerability from other vulnerabilities in a CVE, then these two attributes are all that is necessary. By using only name and description, we reduce the complexity of a Common Vulnerability Enumeration and avoid problems of representation and classification – at least from the operational perspective. Extensions to this limited notion of "vulnerability," using a CVE as the logical bridge, can unite the varying perspectives of what a vulnerability "really" is.

5.3 Conclusions

Schemas and primitive lists attempt to define what a vulnerability is. By attempting to define a robust schema or a long list of primitives for a shareable vulnerability database, we are creating more points over which we can or will disagree and thereby, we create more roadblocks to integration. Moreover, by agreeing to postpone or even eliminate any decision on other shared attributes for a vulnerability entity (beyond name and description), we each achieve a level of freedom to approach, record and manage vulnerability information according to our own needs with no constraint on our modeling imposed on us from other sources. In this way, we can each strive to gain new insights into vulnerabilities from our own perspectives and operational needs, while maintaining the ability to collate and merge our data with others – something we will be able to do if we all agree to link our work to a CVE.

6 Practical Issues

In this section, we will explore a few of the practical matters involved with using a CVE to achieve interoperability between existing systems

6.1 Political Sharability

One big difference between the study of vulnerabilities and the study of species and chemical elements is that the "ownership" of the nomenclature of species and elements has evolved as a part of our language, our culture and as a part of our academic institutions. Vulnerability information, on the other hand, is often tied to commercial interests. For reasons that are not entirely clear, many security organizations do not explicitly enumerate vulnerabilities associated with their own tools (unless those tools are purchased). In other cases where an enumeration is not associated with any tool, the information itself is copyrighted and/or restricted in terms of its distribution; or, the enumeration is largely incomplete, perhaps focusing only on one type of vulnerability.

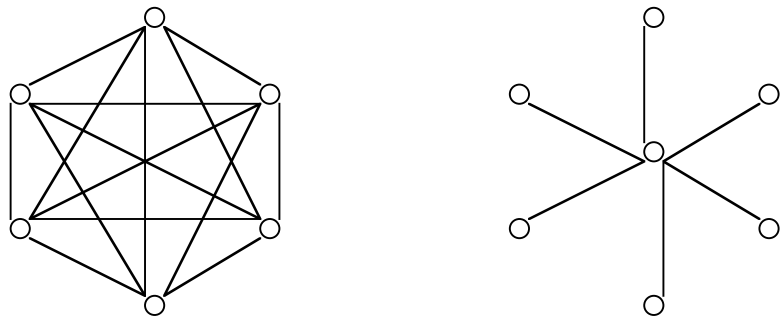
If we are to succeed at defining a CVE, it must be "politically" sharable as well as logically shareable. A collective effort, perhaps by a consortium of commercial, academic, and other interests, may have the most success in establishing a CVE that benefits the community as a whole. Ultimately, however, the definition of a CVE may require a single neutral authority to resolve any conflicts that will arise out of multiple competing interests.

6.2 Reducing Complexity of Mappings

Given two particular databases, the mapping between them must specify which entries in one database are related to entries in the other. For example, if database A enumerates a set of NFS vulnerabilities, and B enumerates its own set, then for each of the vulnerabilities in A, the mapping must specify which of the vulnerabilities in B are related to it. To achieve full interoperability between all databases, we would need to maintain ${}_nC_2 = n\,(n-1)/2$ different mappings.

An alternative to this approach is to introduce an additional database - a CVE - and use it as a standard to which all others are mapped. There are two advantages to this approach.

The first and most obvious is that this reduces the number of mappings to O(n) instead of O(n^2).



a) With no common enumeration.

b) Using a common enumeration.

Managing mappings between databases.

Figure 2

A less obvious advantage may only become realized as a CVE becomes accepted and known by the community. At present, if we want to establish a direct mapping between a particular database that is known and understood and some other database that is presumably not known or understood, we must first acquire a full understanding of how the second database enumerates and names the vulnerabilities. However, if a CVE becomes established and widely accepted by the information security community, it may be easier to map the first database into the CVE simply due to the fact that the CVE is better known and understood than the second database. In this way, mapping each separate database to a CVE restricts the domain of knowledge required to tie a database to other databases.

6.3 Loss of Precision

While mapping a database to a CVE may reduce the overall complexity of the integration of multiple databases, it has an inherent disadvantage: the potential loss of precision. The direct mapping between two databases may be more precise than a composite mapping via the CVE.

We suggest that this potential loss of precision is an acceptable trade-off for two reasons. First, the only alternative is to maintain and share $O(n^2)$ separate mappings. Our experience has been that as a community, we are not currently focusing on this problem. Second, our hope is that more and more vulnerability databases would adopt a standard CVE in their own internal databases. Eventually, this may allow for a greater degree of precision while using composite mappings by way of the CVE, while providing an explicit mechanism for identifying fundamental differences between databases.

6.4 Conclusions

In the effort to achieve interoperability between separate vulnerability databases, linking all databases to a shared CVE reduces the number of mappings between databases from $O(n^2)$ to $O(n)$. This dramatic increase in efficiency may be added to as a CVE gains acceptance in the community. Since the use of a CVE will involve a composite mapping between competing databases, we might expect some loss in precision compared to maintaining separate direct mappings for every pair of databases. However, we believe that this potential shortcoming is outweighed by our need to relate disparate vulnerability databases in a more efficient manner.

7 Future Work

Our current work has been focused on identifying the many information sources we want to integrate into our own vulnerability-focused database. Our experiences have led us to identify the need for a CVE. Future work will entail creating or obtaining a CVE, then using it to assist us in our integration.

Regardless of the CVE we adopt, we anticipate several challenges, most of them related to the general problem of mapping between databases.

While one of the purposes of a CVE is to simplify the mapping between vulnerability databases, we anticipate that the mapping from a database into the CVE will rarely be one-to-one, if ever (consider the NFS example in which one database listed 13 vulnerabilities while another listed 20). Most vulnerability databases are narrow in focus. For example, many are focused only on network-exploitable vulnerabilities (e.g. IDS signature databases) or locally exploitable vulnerabilities (e.g. COPS). Other databases are fairly general but still include only a portion of all the vulnerabilities (e.g. the implicit database of CERT advisories, which only identifies the most high-risk and common vulnerabilities). Each of these databases, due to their purpose, will contain a different enumeration of vulnerabilities, and so will not have a one-to-one relationship with any CVE.

While there is no acceptable CVE for us currently, the anticipated lack of one-to-one mappings from a database to a CVE will be problematic, as there will likely be gaps or differences in the level of abstraction. A mapping that only uses an “equals” relation will be limited in its completeness.

To better clarify the relationships between the elements of a vulnerability database and those of a CVE, we anticipate utilizing the following relations (currently informally defined). Suppose that V1 is a vulnerability as defined in one database, and V2 is a vulnerability that is defined in another database (e.g. a CVE). Then:

V1 = V2	if	(V1 and V2 refer to the same vulnerability)
V1 subsumes V2	if	(V1 includes V2 and other vulnerabilities)
V1 intersects V2	if	(V1 and V2 share some, but not all, characteristics)

The CVE Relations outlined above can be used to more precisely express the relationships between a vulnerability database and the Common Vulnerability Enumeration than just a simple "equals" relation could do. The intersection relation effectively implies that there is some overlap between two vulnerabilities, but they are not equal.

These CVE relations can best be exhibited using a sample CVE consisting of a subset of the known NFS vulnerabilities:

Name	Description
CE1	NFS file handle guessing
CE2	NFS pcnfsd
CE3	NFS world export
CE4	NFS export list > 256 char

We can use the CVE Relations to map from CERT advisories (CA), CyberCop Scanner probes (CCS), and X-Force database entries (XF) to those vulnerabilities specified in our CVE:

CyberCop Scanner	ISS X-Force	CERT	CVE
CCS 7009 = CE1	XF 77 = CE1	CA-91.21 subsumes CE1	CE1
CCS 6004 = CE2	XF 415 = CE2	CA-96.08 = CE2	CE2
CCS 7002 = CE3	XF 798 = CE3	(No CERT is related to CE3)	CE3
CCS 7013 = CE4	XF 503 = CE4	CA-94.02 subsumes CE4	CE4

7.1 Using CVE Relations to map between vulnerability databases

If we define a vulnerability as a set of (arbitrarily defined) attributes and interpret the CVE relations accordingly, then "equals" and "subsumes" have the following properties:

$$(V1 = V2) \Rightarrow V2 = V1$$
$$(V1 = V2 \text{ and } V2 = V3) \Rightarrow V1 = V3$$
$$(V1 \text{ subsumes } V2) \text{ and } (V2 \text{ subsumes } V3) \Rightarrow V1 \text{ subsumes } V3$$

Unfortunately, the transitive property does not hold for intersection. However, it can still be used to determine potential interactions between databases:

$$(V1 \text{ intersects } V2) \text{ and } (V2 \text{ intersects } V3) \Rightarrow V1 \text{ might intersect } V3$$

The transitive property of these relations might be used to make inferences such as:

(CCS 7002 = CE3) and (XF 798 = CE3) $\Rightarrow$ CCS 7002 = XF 798
(CERT CA-91.21 subsumes CE1) & (CE1 = XF 77) $\Rightarrow$ CERT CA-91.21 subsumes XF77.

7.2 Measuring the quality of a Candidate CVE using its CVE Relations

The CVE Relations might be able to be used to objectively measure the quality of a candidate CVE that is being considered for adoption, or to compare potentially competing CVE's. We might evaluate an enumeration E with respect to a vulnerability database D by measuring the cardinality of the derived relations and defining appropriate metrics. One approach would be to choose a CVE which maximizes the number of "equals" relationships and minimizes the number of intersections (as intersections could imply vague relationships). E's specificity might be measured using the cardinality of the "subsumes" relation from E to D, and E's generality might be measured with the cardinality of the subsumes relation from D to E. Quality metrics might also include measurements such as the number of gaps in E with respect to D, i.e. vulnerabilities in D which have no relationship to any vulnerabilities in E.

8 Conclusions

We believe that the information security community needs to construct and adopt a shareable vulnerability database that captures an agreed upon Common Vulnerability Enumeration. A simple enumeration of the known vulnerabilities is badly needed in order to achieve interoperability of security tools and databases and to foster communication in the information security community. Further, we believe that a Common Vulnerability Enumeration of this type is achievable right now. We propose that a good first attempt with regard to determining the proper level of abstraction and the proper enumeration of the vulnerabilities can be achieved by examining the current major commercial and publicly available vulnerability databases. Since there is currently no consensus with regards to the definition of a robust schema or a list of primitives, we propose that a shareable database should strive for no more than the definition of a name and description for each vulnerability. Since a Common Vulnerability Enumeration must be politically shareable as well as logically shareable, we propose that such an enumeration of vulnerabilities should eventually be derived and maintained by a consortium of parties or by some commercially neutral party.

References

[1] ASLAM, T. 1995. A Taxonomy of Security Faults in the Unix operating system. M.S. thesis, Purdue University.

[2] BISHOP, M. AND BAILEY, D. 1996. A Critical Analysis of Vulnerability Taxonomies. Tech. Rep. CSE-96-11, Department of Computer Science at the University of California at Davis. September.

[3] CERT COORDINATION CENTER. 1991. CERT Advisory CA-91:21. Published electronically at http://www.cert.org/ftp/cert_advisories/CA-91%3a21.SunOS. NFS.Jumbo.and.fsirand

[4] ENCYCLOPEDIA AMERICANA INTERNATIONAL EDITION. 1991. Volume 3. Bear. Grolier Incorporated.

[5] ENCYCLOPEDIA AMERICANA INTERNATIONAL EDITION. 1991. Volume 26. Taxonomy. Grolier Incorporated.

[6] INFILSEC SYSTEMS SECURITY. 1997. Online Database. Published electronically at http://www.infilsec.com/.

[7] INTERNET ENGINEERING TASK FORCE. 1998. Intrusion Detection Exchange Format Charter. Published electronically at: http://www.ietf.org/html.charters/idwg-charter.html

[8] INTERNET SECURITY SERVICES. 1999. Online Database X-Force. Published electronically at http://www.iss.net/xforce/.

[9] NATIONAL INFORMATION ASSURANCE PARTNERSHIP. 1998. Online Database Catalog of Threats and Countermeasures. Published electronically at http://niap.nist.gov/TnC%20HTML%201/Index.html

[10] NETWORK ASSOCIATES INCORPORATED. 1999. Proprietary Vulnerability Database for CyberCop Scanner 2.4.

[11] SCERRI, E. September, 1998. The Evolution of the Periodic System. Scientific American.