LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TH-208822

Software Vulnerability Taxonomy Consolidation

Sriram Polepeddi

January 4, 2005

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Software Vulnerability Taxonomy Consolidation

Masters Thesis

Sriram S. Polepeddi

December 7, 2004

Information Networking Institute
Electrical & Computer Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In today's environment, computers and networks are increasing exposed to a number of software vulnerabilities. Information about these vulnerabilities is collected and disseminated via various large publicly available databases such as BugTraq, OSVDB and ICAT. Each of these databases, individually, do not cover all aspects of a vulnerability and lack a standard format among them, making it difficult for end-users to easily compare various vulnerabilities. A central database of vulnerabilities has not been available until today for a number of reasons, such as the non-uniform methods by which current vulnerability database providers receive information, disagreement over which features of a particular vulnerability are important and how best to present them, and the non-utility of the information presented in many databases.

The goal of this software vulnerability taxonomy consolidation project is to address the need for a universally accepted vulnerability taxonomy that classifies vulnerabilities in an unambiguous manner. A consolidated vulnerability database (CVDB) was implemented that coalesces and organizes vulnerability data from disparate data sources. Based on the work done in this paper, there is strong evidence that a consolidated taxonomy encompassing and organizing all relevant data can be achieved.

However, three primary obstacles remain: lack of referencing a common ‘primary key’, un-structured and free-form descriptions of necessary vulnerability data, and lack of data on all aspects of a vulnerability. This work has only considered data that can be unambiguously extracted from various data sources by straightforward parsers. It is felt that even with the use of more advanced, information mining tools, which can wade through the sea of unstructured vulnerability data, this current integration methodology would still provide repeatable, unambiguous, and exhaustive results. Though the goal of coalescing all available data, which would be of use to system administrators, software developers and vulnerability researchers is not yet achieved, this work has resulted in the most exhaustive collection of vulnerability data to date.

Acknowledgments

I gratefully acknowledge the tremendous enthusiasm and guidance given by Terry Brugger, my supervisor at Lawrence Livermore National Labs, and by Dr. Tom Longstaff, my advisor from the Computer Emergency Readiness Team and the Software Engineering Institute. Many times, Terry came to the rescue with the equipment, resources, and ideas necessary to help this work proceed smoothly. He always had the right suggestion at the right time.

Dr. Longstaff provided many great suggestions on which avenues should best be pursued for this work as well as very early warning of potential dead-ends. Despite his very busy schedule, he made himself available to me every time I had questions or ideas for him.

In addition, I would like to thank Noel Tijerino, my co-supervisor at LLNL, for always helping to sort out the various issues that would arise during the implementation of this work. I would also like to thank the IOAC at LLNL for providing any and all necessary resources for carrying out this work. I have benefited from many discussions with members of the IOAC and am deeply humbled by the range understanding each of them have of this field.

Lastly, I would like to thank my Father for his motivation and support.

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48. UCRLxx-xxxxxxx.

Table of Contents

List of Tables

List of Charts

1. Background

Today's computers and networks are plagued by an increasing number of software vulnerabilities. The heterogeneity of vulnerable software and the multitude of providers of error-ridden software results in a myriad exploits by which computers can be compromised. However, while information about these vulnerabilities is collected and disseminated via various large publicly available databases such as BugTraq, OSVDB, ICAT, etc., it is neither centrally accessible nor presented in a standardized manner, making it difficult for end-users and administrators to easily combine and compare various vulnerabilities [HL98].

There are many reasons why such a central database of vulnerabilities as not been available till today.

1. Non-uniform methods by which current vulnerability database providers receive information from vendors, white-hat hackers, and the public, requiring the modifying of each input to their particular database schema.
2. General disagreement over which features of a particular vulnerability are important and how best to present them.
3. The non-utility of the information presented in many databases. Most existing databases simply list information describing the vulnerability but fail to deal with the question of how what to do next, in a structured manner, let alone the larger issues of how important it is to resolve the vulnerability, or even the un-intended consequences of installing a faulty patch.
4. No open-standard non-government owned database until the Open Source Vulnerability Database opened to the public in April 2004. This database is however missing many aspects of a vulnerability in favor of a streamlined data-handling approach. The most up-to-date database, BugTraq, is owned by Symantec, a private interest, and other government Databases by CERT, CIAC and NIST (ICAT) are slower to add new vulnerabilities.

The solution lies in a universally accepted vulnerability taxonomy that classifies the vulnerabilities in a manner that allows for unambiguous storage and classification. Any such taxonomy must satisfy the following 6 criterion [HL98]:

1	2	3	4	5	6
Mutual Exclusivity	Exhaustiveness	Unambiguity	Repeatability	Acceptability	Utility

Table 1: Six Criteria for a Satisfactory Taxonomy

Such a taxonomy would alleviate the first, second and third obstacles to a universally accepted vulnerability database. Many databases satisfy criterions 1, 3, 4, yet are not exhaustive enough to achieve acceptability and do not deal with the question of what next, to be fully usable.

2. Problem Statement

The lack of a consolidated source of vulnerability information requires software developers, vulnerability researchers and system administrators to individually identify and bring together vulnerability information relevant for their work. Software developers need to know which vulnerabilities, either caused by their own poor design choices or use of off-the-shelf packages and libraries, to avoid in future releases. Vulnerability researchers often need access to data on the frequency, origin, location, scope, target and solutions to vulnerabilities [LGH96]. And, system administrators must know what holes exist in their network as well as the potential sources of attack as soon as this information is available. A consolidated vulnerability database would prevent each of these users from duplicating the effort necessary to gather relevant vulnerability information, as well as, satisfy the main requirements of each [OSVAIM].

3. Thesis Statement

By empirically examining currently available vulnerability data sources, it is possible to construct a comprehensive and consolidated taxonomy of currently existing vulnerability information. Each data source containing a particular feature of a vulnerability will be evaluated versus four of the six taxonomic criteria listed above. The acceptability and utility of these features will be discussed in on a per data source basis in Section 7.3 and only for the consolidated database in Section 8.2. These individual features will be combined into a single schema. Then, each consolidated data set will be compared to the original to show that it maintains the same properties, and therefore that the 6 criteria above [HL98] still hold. Once the database is created, its scalability will be evaluated. It is assumed that a front-end to take advantage of this database will be developed by anyone requiring use of the data contained herein.

4. Previous Work

Much work has been done in the past two decades regarding the nature of vulnerabilities and the ideal method of classifying them. Initial work in pure vulnerability classifications centered on fitting a vulnerability into 7-10 flat categories. [PA, RISOS] These often resulted in categories which were ambiguous, as they did not have a well-defined notion of a software vulnerability. Landwher et al. [L93] proposed a taxonomy based on genesis, time of introduction and location of its introduction. Unfortunately, this scheme requires *a priori* knowledge of the vulnerability's programmer's intent and of the software design lifecycle phase where it was introduced.

A clear evolution can be seen moving from simpler classification methods to more elaborate schemas. Aslam's Classification of UNIX Security faults [AS95, AS96] presents a highly focused fault taxonomy. It categorized 49 CERT advisories as either an operational, environmental, or coding fault. These were then further divided into 2 further levels resulting in 14 leaf nodes. [BB96] have shown however that it is possible to classify a fault (such as the xterm log file flaw) in multiple categories; therefore, this classification is neither mutually exclusive nor repeatable. Also, given the limited number of flaws categorized, it cannot necessarily claim to be exhaustive. In addition, it does not support categorization of the location of the fault (network component, kernel, application, etc.) or the type of access needed

(physical, local, remote/network) to exploit the vulnerability. It also does not address the impact of a vulnerability.

Perhaps the best work to date on the subject was done by Ivan Krsul, which builds on earlier work by Aslam [AS95], and includes categories for "Access Required" and "Ease of Exploit" (based on [TAL97]), and "System Component" (based on [TB87]). This taxonomy does however have some redundant information, categorizing a flaw based on its ease of exploit and its complexity to exploit separately. Also, it does not accurately deal with the question of what to do once a vulnerability has been determined to exist on a particular system. Furthermore, the unavailability of most of the data sources used (INFILSEC, Michael Dilger's Vulnerability DB, Eric Miller's Vulnerability DB, the AFIW's CMET DB, and Mike Neuman's DB) make reproducing this work impracticable. Nevertheless, the schema envisioned by Krsul does handle most of the basic requirements of a vulnerability taxonomy, and can be fine-tuned to provide a more comprehensive taxonomy.

Current implementations of vulnerability databases all center around a common theme of listing vulnerabilities, assigning them unique IDs, along with descriptions, affected components, and potential fixes. However, many give a very cursory description of the severity of the problem, potential ways in which the vulnerability may be exploited, and only textual descriptions of the solution making it difficult to unambiguously glean data that can be consolidated into a data repository.

5. Methodology

The goal of this work is to evaluate the feasibility of consolidating the major white-hat vulnerability data sources in a manner that allows the greatest number of Full Records. Full Records in this work are those records that contain the following information on a vulnerability:

- Category
- Dates of Vulnerability Disclosure/Discovery/Exploit Release
- Severity

Three major vulnerability Data sources, BugTraq, OSVDB, and ICAT can be taken as the foundation upon which to attach ancillary information, such as Ease of Exploit and Solution Status. The work of consolidating available data into an *a posteriori* taxonomy will be done in the following phases.

1. Study the currently available data sources to understand what information is available and select data sources with useful and linkable information to be consolidated.
2. Consider the Determine the ideal primary key for linking the disparate data sources.
3. Examine each of the chosen data sources for the best data on various features of a vulnerability. "best" refers to how well the data maps to a suitable taxonomy as described in [HL98]. As each chosen data source is integrated into CVDB, the distributions within the original data will be compared to the consolidated data to reveal how well existing information was mapped.

The evaluation of this new taxonomy would be based primarily on its exhaustiveness and utility in addition to the other principles. The data set produced by this proposed taxonomy will be compared to each of the individual data sources.

6. Environment

This project will be implemented at the Department of Energy's Lawrence Livermore National Labs (LLNL) in its Information Assurance Operations Center (IOAC) in Livermore, California. This will extract information from web-based data sources and non-web based formats such as XML and CSV for use in a consolidated schema. The analysis of the work will be completed at the Information Networking Institute at Carnegie Mellon University, Pittsburgh.

7. Implementation

7.1. Phase 1: Survey of Currently Available Data Sources

Currently available vulnerability data sources were examined for the following vulnerability features.

Data Source	**Chosen**					**Not Chosen**		
Vulnerability Features	**BugTraq**	**ICAT**	**OSVDB**	**Computer Associates' VIC**	**Secunia**	**CERT**	**ISS X-Force**	**Cerias's CoopVDB**
Category	X	X	X					
Loss Type		X	X					
Affected Component		X						
Attack Location	X	X	X	X	X			
Severity		X		X	X			
Exploit Availability			X					
Ease of Exploit					X			
Solution Status	X				X	X	X	
Software Information	X	X	X	X	X	X	X	
Disclosure Date			X			X	X	
Discovery Date			X	X				
Exploit Release Date			X					
Up-to-date?	**Yes**	**Mostly**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**No**

Table 2 Features Available on Current Vulnerability Databases

The following databases were chosen and will be referenced often throughout this work. The information referenced from each database can be found at the following locations:

- BugTraq: http://www.securityfocus.com/bid/cveid/
- ICAT: http://icat.nist.gov/icat.cfm/
- OSVDB: http://www.osvdb.org/search.php
- CA's VIC: http://www3.ca.com/securityadvisor/vulninfo/
- Secunia: http://secunia.com/historic_advisories/

Full Descriptions of the data sources not chosen for consolidation are given in Appendix I.

7.2. Phase 2: Primary Key Selection

A primary key is necessary to link the various data sources together and will be chosen from the currently available data sources based on the following criteria:

1. Acceptability: The number of other sources/products that refer to it.
2. Total links to other data sources
3. Total vulnerabilities represented
4. Link/Vulnerability Ratio: If a particular vulnerability ID were to be used as a primary key, it would need have references to more than one external source, the higher this ratio, the greater the amount of data that can be integrated.

Note that the choice of primary key is not dependent on the amount of information that is linked in its native data source. This is because our goal is to evaluate how well we can connect the various disparate data sources. How well individual records were linked will be evaluated in Section 8.1.1.

Another factor considered important was the up-to-datedness of the vulnerability source, yet as most chosen sources had updated entries listed within the last few days, and since the spacing between vulnerability releases was never more than a few days, this criteria was not used when choosing a primary key.

Candidates for primary key are:

- Bugtraq ID
- CVE ID
- Computer Associates' Vulnerability Information Center ID
- OSVDB ID

The ICAT ID was not considered as a primary key for this database as it uses CVE IDs as its ID.

Bugtraq ID:

1. Acceptability: While BIDs are well-known, no list products asserting BugTraq-compatibility was found.
2. Total Links to other data sources: = 23405
3. Total Vulnerabilities represented: 10913
4. Reference/Link ratio: 23405/10913 = **2.14**

CA VIC's ID

1. Acceptability: Currently, 3 Computer Associates products refer to these IDs.
2. Total Links to other data sources: 10382
3. Total vulnerabilities represented: 8982
4. Reference/Link ratio: 10382/8982 = **1.16**

CVE ID

1. Acceptability: Currently 200 products and services from 125 vendors are CVE-Compatible [CVE FAQ].
2. Total Links to other data sources: 38362
3. Total vulnerabilities represented: 7532
4. Reference/Link ratio: 38362/7532 = **5.09**

OSVDB ID

1. Acceptability: Currently, OSVDB IDs are supported by 3 open-source products.
2. Total Links to other data sources: 22345
3. Total vulnerabilities represented: 5335
4. Reference/Link ratio: 22345/5335 = **4.19**

Choice of Primary Key

CVE has the largest total number of external references (38362), the highest Reference/Link Ratio (5.09), the 3rd largest vulnerabilities represented (7532) and is overwhelmingly supported by the security industry.

7.3. Phase 3: Data Source Selection for Each Vulnerability Feature

The tables below will evaluate each of the data sources that contain vulnerability feature information against these four of the six criteria for a taxonomy.

- Mutual Exclusivity
- Exhaustiveness
- Unambiguity
- Repeatability

The acceptability of the data sources can be prioritized in the following order.

1. BugTraq, by far the most accepted database among these five.
2. ICAT, open, better than OSVDB due to its strong association with CVE.
3. OSVDB, open, might overtake ICAT in the future, already supported by three network security products.
4. CA VIC, proprietary, recording vulnerabilities since 1998
5. Secunia, proprietary, recording vulnerabilities since 2002

The last criteria, utility will be shown for the overall database in Section 8.2.

Whenever possible, the specific fields from a data source considered for a particular feature have been listed. When information from a data source is available via multiple sources, such as the web, XML exports or CSV files, the field name was chosen from the web-based source. Explanatory notes were given for data sources not chosen, though containing relevant information on a particular feature.

After each table, a chart of the distributions of each feature in the original data source vs. CVDB is given.

Vulnerability Categories:

Criteria	BugTraq	ICAT	OSVDB
Field(s) Used	**Class**	**Vulnerability Type**	**Attack Type**
Mutual Exclusivity	No vulnerability is present in multiple groups.	Vulnerabilities are present in multiple categories.	A few vulnerabilities are present in multiple categories.
Exhaustiveness	100% of 10913 vulnerabilities classified in 10 groups.	94.12% of 6993 of vulnerabilities classified in 9 groups.	77.17% of 5335 vulnerabilities are classified in 9 categories. Vulnerabilities in the Other or Unknown groups were not included.
Unambiguity	Combines buffer overflows and boundary conditions together in a single category. Also, boundary Condition Errors can be considered a subset of Input Validation Errors, yet they are listed separately here leading to some ambiguity.	Handles the Input Validation (IVE) → Boundary Condition Error (BCE) → Buffer Overflow (BO) issue better than BugTraq, but not perfectly. A buffer overflow can be a type of boundary condition verification error, and a boundary condition verification error can be a type of Input Validation Error. ICAT is able to show that IVEs are a super-set of BCEs and BOs. However, it fails to show that BCEs are a super-set of BOs.	No criteria for placing a vulnerability in a particular group is given. OSVDB only presents lists of generic manifestations of a category as a guide.
Repeatability	Other than the BO/BCE issue, there is not much subjectivity in deciding which category to place a vulnerability.	The definitions used for each of these categories seem precise enough for any classifier to arrive at the same results, though this may result in a vulnerability appearing in multiple categories.	It is possible that different classifiers may place a vulnerability in different categories.
Data Source Chosen	**BugTraq**: While it may be possible for a vulnerability to be classified in multiple categories, the aim of most previous vulnerability categorization taxonomies has been to achieve mutual exclusivity, and this is something BugTraq achieves better than the others. BugTraq is also more exhaustive than the others. With respect to Unambiguity and repeatability, BugTraq and ICAT are even.		

Table 3: Category Feature Data Source Evaluation

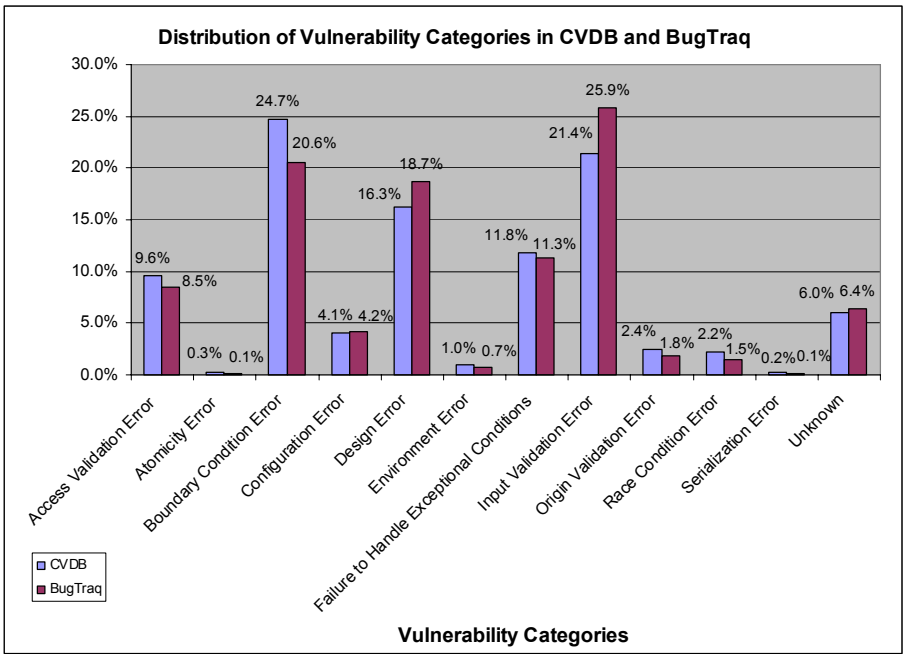

Chart 1: Distribution of Vulnerability Categories in CVDB and BugTraq

The chart above shows the percent of vulnerabilities distributed in each of the *Category* groups in CVDB and BugTraq, with a variance of: **0.0412%**

Loss Type:

Criteria	OSVDB	ICAT
Field(s) Used	**Impact**	**Loss Type**
Mutual Exclusivity	Vulnerabilities are present in multiple categories.	Vulnerabilities are present in multiple categories.
Exhaustiveness	78.65% of 5335 vulnerabilities are classified in at least 1 of 3 groups. Vulnerabilities included in the Unknown group have not been included as they give no additional information about a vulnerability.	96.75% of 6993 of vulnerabilities classified in at least 1 of 4 groups.
Unambiguity	The definitions given for each Impact group are short, yet clear. However, it does lead to vulnerabilities which span groups.	The definitions given for each Loss Type group are clear, yet lead to vulnerabilities which span groups.
Repeatability	The definitions used for each of these groups seem precise enough for any classifier to arrive at the same results, though this may result in a vulnerability appearing in multiple categories.	The definitions used for each of these groups seem precise enough for any classifier to arrive at the same results, though this may result in a vulnerability appearing in multiple categories.
Data Source Chosen	**ICAT**: While similar in most respects, ICAT contains a greater number of vulnerabilities and classifies a greater percentage of them than OSVDB.	

Table 4: Loss Type feature Data Source Evaluation

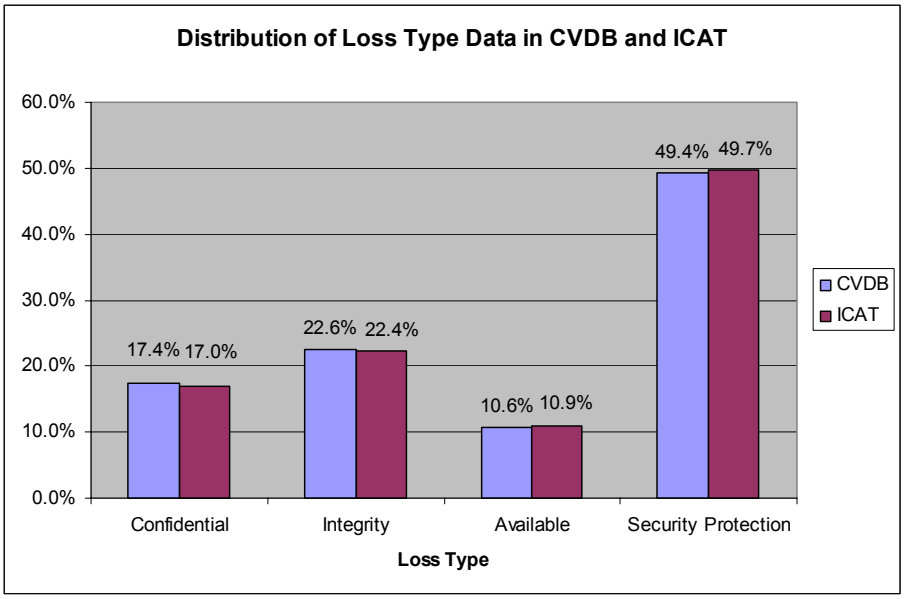

Chart 2: Distribution of Loss Type Data in CVDB and ICAT

The chart above shows the percent of vulnerabilities distributed in each of the *Loss Type* groups in CVDB and ICAT, with a variance of: **0.0719%**

Affected Component:

Criteria	ICAT
Field(s) Used	**Exposed System Component**
Mutual Exclusivity	Vulnerabilities are present in multiple categories.
Exhaustiveness	94.59% of 6993 of vulnerabilities classified in at least 1 of 7 groups.
Unambiguity	Only groups names, 'Server application', 'Hardware', etc. are given, yet these are generally well-known, so they need no further explanation. Possible sources of ambiguity lie between groups such as 'Protocol stack' and 'Communication protocol'. An error in a stack could possibly occur in a particular communication protocol alone; no clear delineation is given between these.
Repeatability	Other than the ambiguity above, the group names are precise enough for any classifier to arrive at the same results.
Data Source Chosen	**ICAT**: The only source which contained this information in a structured manner.

Table 5: Affected Component Feature Data Source Evaluation

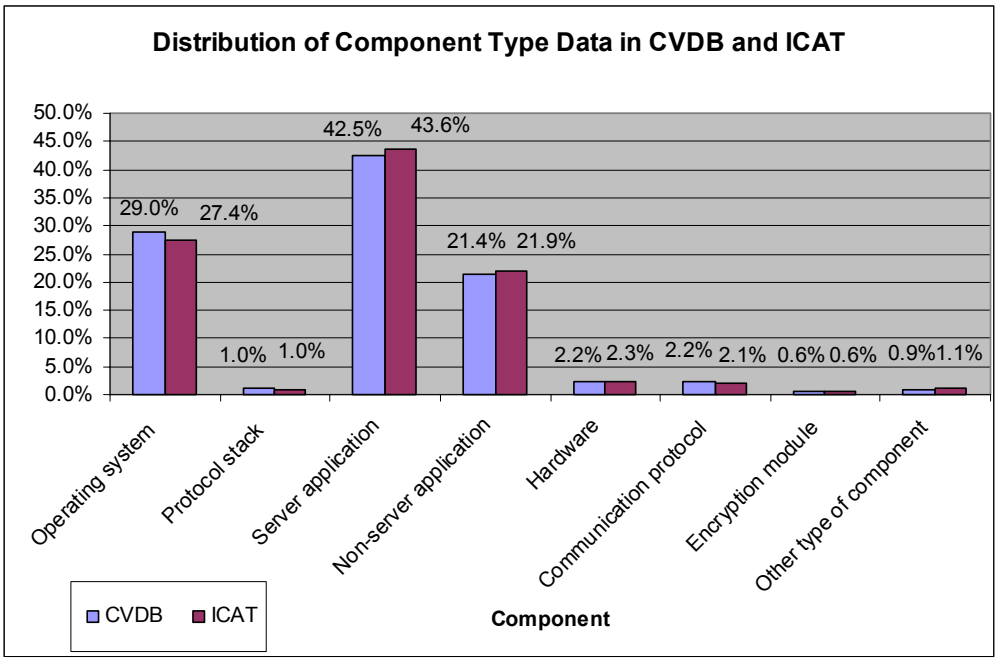

Chart 3: Distribution of Component Type Data in CVDB and ICAT

The chart above shows the percent of vulnerabilities distributed in each of the *Affected Component* groups in CVDB and ICAT, with a Variance of: **0.0062%.**

Attack Location:

Criteria	OSVDB	BugTraq	ICAT
Field(s) Used	**Console/Physical, Shell/Local, Network/Remote, Telephony, Unknown**	**Local, Remote**	**Exploitable Range**
Mutual Exclusivity	Vulnerabilities are present in multiple categories.	Vulnerabilities are present in multiple categories.	Vulnerabilities are present in multiple categories.
Exhaustiveness	80.19% of 5335 vulnerabilities are classified in at least 1 of 4 groups. Vulnerabilities included in the Unknown group have not been included as they give no additional information about a vulnerability.	98.96% of 10913 vulnerabilities classified in 2 groups. Some vulnerabilities contain the value Unknown and have not been included.	95.72% of 6993 of vulnerabilities classified in at least 1 of 2 groups.
Unambiguity	The definitions given for most groups are clear. However, the telephony group may also include vulnerabilities which fall in the network/remote or shell/local.	There is a clear delineation between a local or remote attack.	There is a clear delineation between a local or remote attack.
Repeatability	Though ambiguous, the definitions used for each of these groups are precise enough for any classifier to arrive at the same results.	The definitions used for each of these groups seem precise enough for any classifier to arrive at the same results.	The definitions used for each of these groups seem precise enough for any classifier to arrive at the same results.
Data Source Chosen	**ICAT**: Though OSVDB classifies vulnerabilities into a greater number of attack location groups, its classification is somewhat ambiguous. In all other respects as well, BugTraq and ICAT's classification is better. Though BugTraq classifies the greatest percentage of its vulnerabilities in this feature, it contains some fields with unknown values. Therefore ICAT which classifies a similar percentage of its vulnerabilities with no unknown values was chosen.		

Table 6: Attack Location Feature Data Source Evaluation

Sources not considered: Though Secunia and CA VIC contain attack location data, they were not considered for this feature as they clearly contained a small amount of integrate-able data, and the required information was available via other sources.

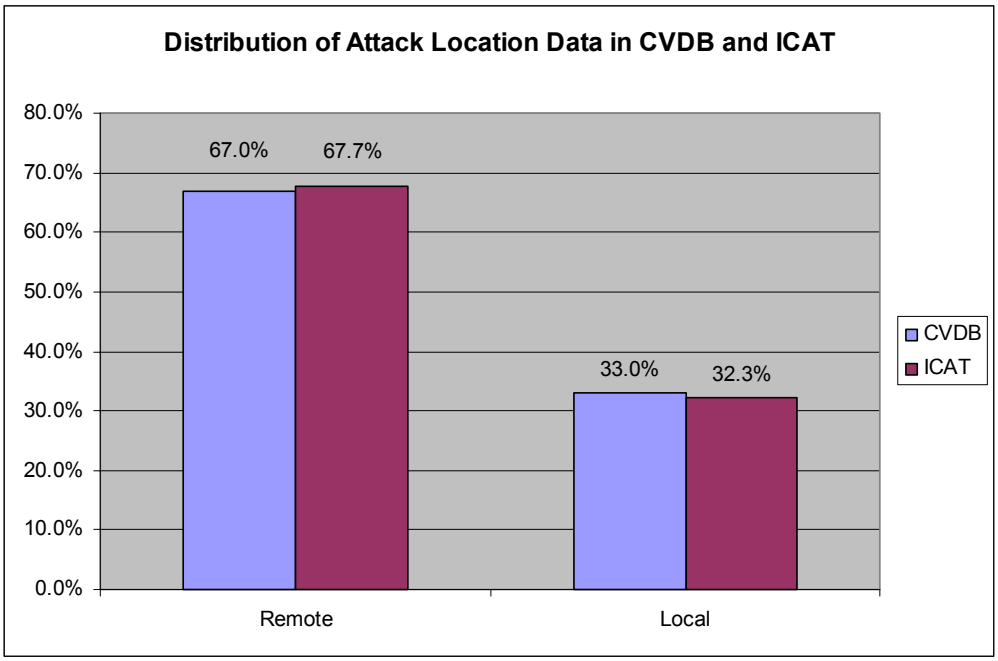

Chart 4: Distribution of Attack Location Data in CVDB and ICAT

The chart above shows the percent of vulnerabilities distributed in each of the *Attack Location* groups in CVDB and ICAT, with variance of: **0.0112%**

Severity:

Criteria	Secunia	CA VIC	ICAT
Field(s) Used	**Critical**	**Threat Impact**	**Severity**
Mutual Exclusivity	Vulnerabilities are present in only 1 group.	Vulnerabilities are present in only 1 group.	Vulnerabilities are present in only 1 group.
Exhaustiveness	100% of 5967 vulnerabilities are classified in 1 of 5 groups.	100% of 8982 vulnerabilities are classified in 1 of 5 groups.	96.95% of 6993 vulnerabilities are classified in 1 of 3 groups
Unambiguity	The definitions given for each Critical group are clear.	The definitions given for each Threat Impact group are clear.	The delineations between the groups are not precise.
Repeatability	The definitions used for each of these groups seem precise enough for any classifier to arrive at the same results.	The definitions used for each of these groups, though clear, require a bit of subjectivity and may not lead to a classifier arriving at the same results.	As the definitions are not clear and require a great deal of subjectivity, a classifier will likely not arrive at the same results.
Data Source Chosen	**Secunia**: Though Secunia contains the smallest vulnerability set, it classifies them better than the other two.		

Table 7: Severity Feature Data Source Evaluation

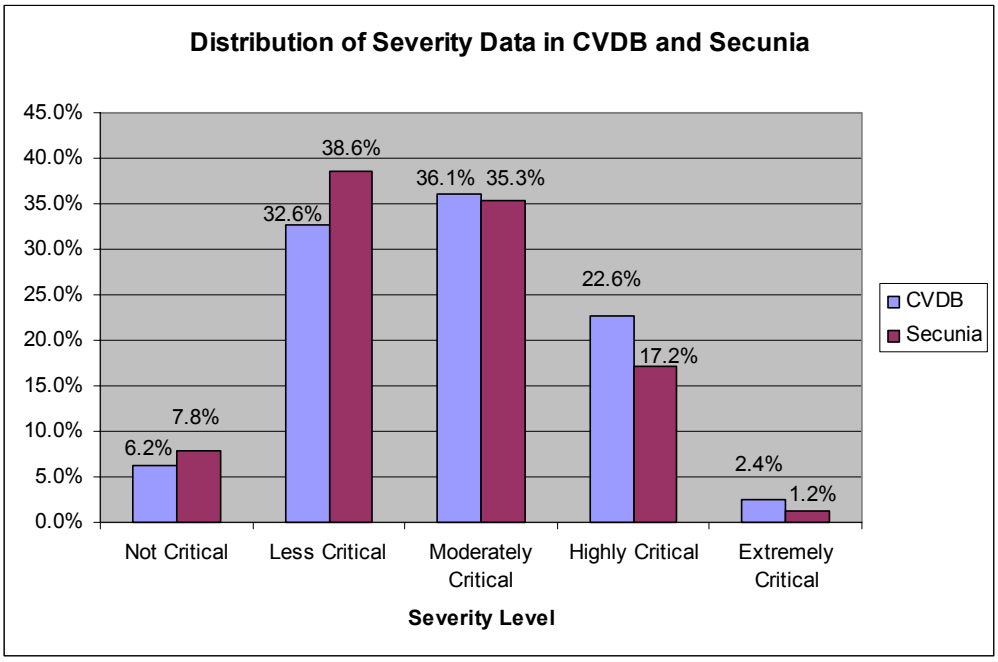

Chart 5: Distribution of Severity Data in CVDB and Secunia

The chart above shows the percent of vulnerabilities distributed in each of the *Severity* groups in CVDB and Secunia, with variance of: **0.1726%**

Exploit Availability:

Criteria	OSVDB
Field(s) Used	**Exploit**
Mutual Exclusivity	Vulnerabilities are not present in multiple categories.
Exhaustiveness	81.69% of 5335 vulnerabilities are classified in 1 of 4 groups. Vulnerabilities included in the Unknown group have been included as it is likely that an exploit exist which is not known.
Unambiguity	The definitions given for each Exploit group are clear.
Repeatability	The definitions used for each of these groups seem precise enough for any classifier to arrive at the same results, and there is little chance of a vulnerability appearing in multiple categories.
Data Source Chosen	**OSVDB**: The only source which classified this feature in a structured and un-subjective manner.

Table 8: Exploit Availability Feature Data Source Evaluation

Sources not considered: BugTraq contains links to exploits on a separate webpage, but it does not store the information in a structured manner. CA VIC contains a field, *Threat Popularity*, however it requires a subjective evaluation by a classifier.

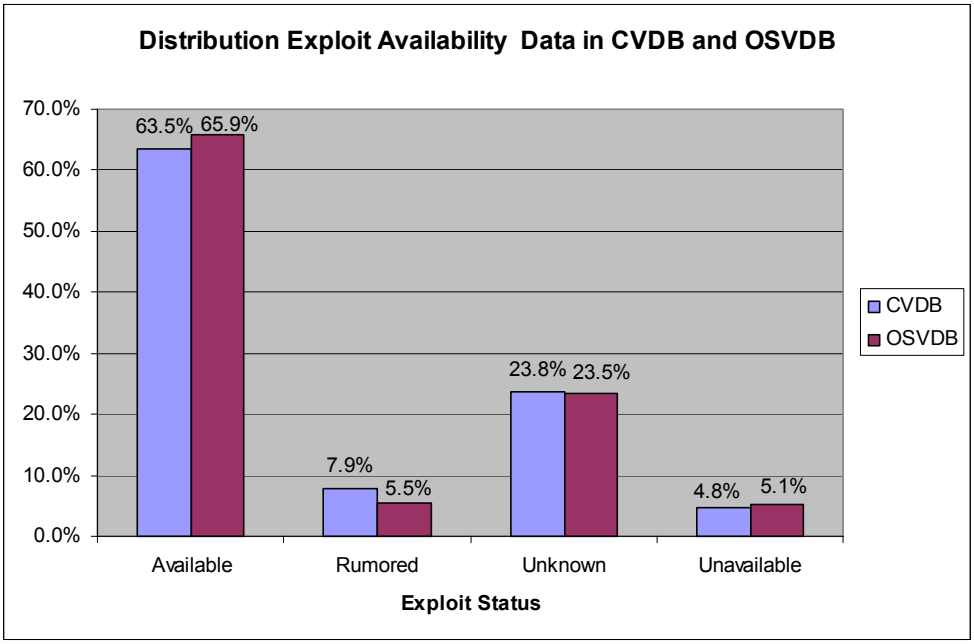

Chart 6: Distribution of Exploit Data in CVDB and OSVDB

The chart above shows the percent of vulnerabilities distributed in each of the *Exploit Availability* groups in CVDB and OSVDB, with variance of: **0.0386%**

Ease of Exploit:

Criteria	CA VIC
Field(s) Used	**Threat Simplicity**
Mutual Exclusivity	Vulnerabilities are not present in multiple groups.
Exhaustiveness	100% of 8982 vulnerabilities are classified in 1 of 5 groups.
Unambiguity	The definitions given for each Threat Simplicity group are clear.
Repeatability	The definitions used for each of these groups, though clear, require a bit of subjectivity and may not lead to a classifier arriving at the same results.
Data Source Chosen	**CA VIC**: Though some subjectivity exists, this was the only source which classified this feature in a structured manner.

Table 9: Ease of Exploit Feature Data Source Evaluation

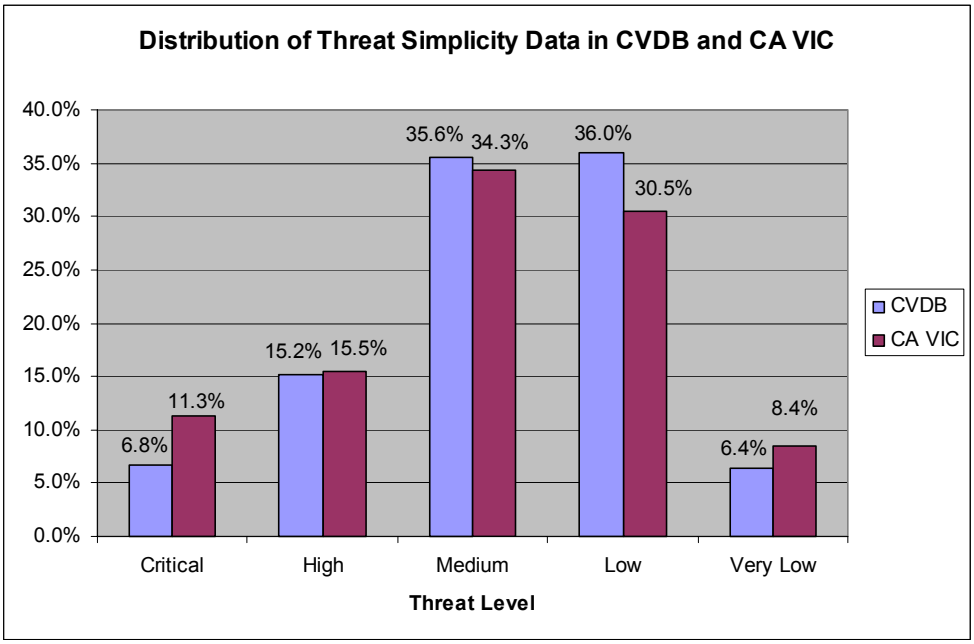

Chart 7: Distribution of Threat Simplicity Data in CVDB and CA VIC

The chart above shows the percent of vulnerabilities distributed in each of the *Ease of Exploit* groups in CVDB and CA VIC, with variance of: **0.1397%**

Solution Status:

Criteria	Secunia
Field(s) Used	**Solution Status**
Mutual Exclusivity	Vulnerabilities are not present in multiple groups.
Exhaustiveness	99.97% of 5967 vulnerabilities are classified in 1 of 4 groups.
Unambiguity	The definitions given for each Solution Status group are clear.
Repeatability	The definitions used for each of these groups seem precise enough for any classifier to arrive at the same results.
Data Source Chosen	**Secunia**: The only source which contained this information in a structured manner.

Table 10: Solution Status Feature Data Source Evaluation

Sources not considered: BugTraq contains links to solutions on a separate webpage, but it does not store the information in a structured manner.

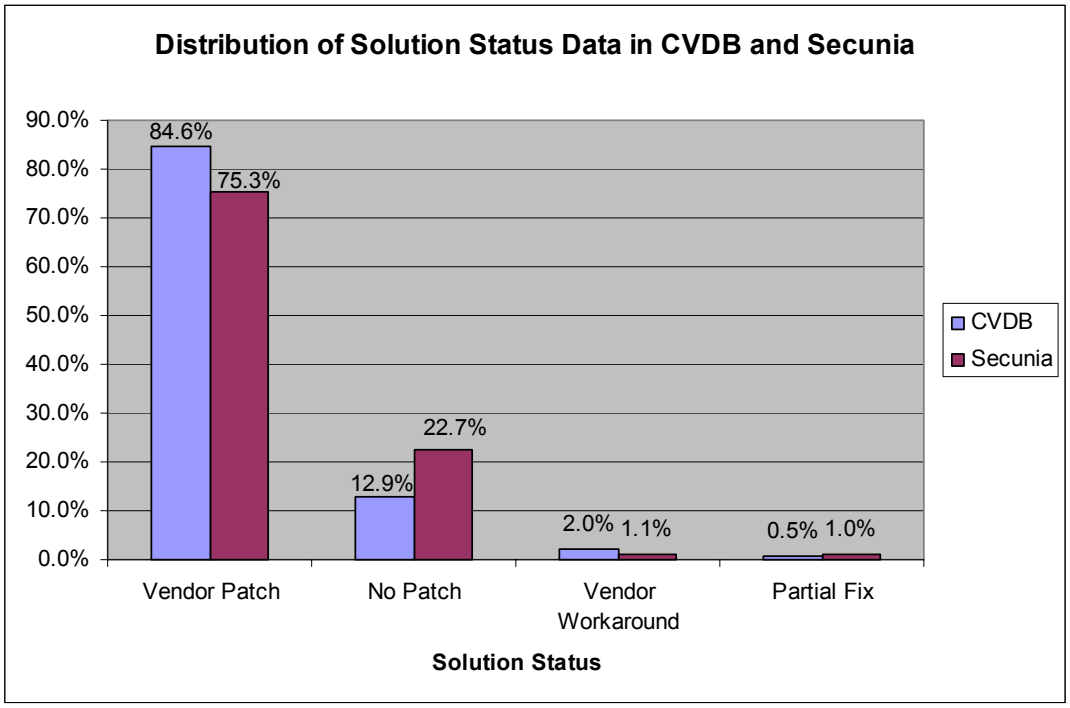

Chart 8: Distribution of Solution Status Data in CVDB and Secunia

The chart above shows the percent of vulnerabilities distributed in each of the *Solution Status* groups in CVDB and Secunia, with variance of: **0.6077%**. This is likely due to incomplete reference to CVE ID.

Software Information:

This relates to data on the platforms and applications which may or may not be affected by a particular vulnerability. It is not clearly a *feature* of a vulnerability, but is nevertheless, necessary information pertaining *to* a vulnerability. There are no set guidelines for what software information is attached to a vulnerability by any of the major databases, therefore, the repeatability and unambiguity criterion cannot be judged. Likewise, mutual exclusivity is not an issue here, except that software cannot be listed as both vulnerable and not vulnerable; no software falls in both categories. So the choice of data source to link to for software information will be made based on the number of software records linked to overall, as well as the level of granularity present in the software record. Ideally, a software record should be easily separate-able into three components: VENDOR : PRODUCT : VERSION

ICAT: This database stores only vulnerable software, and provides for a way to easily separate the software record into its three components. It currently contains 15893 unique vulnerable software records.
BugTraq: This database stores both vulnerable and non-vulnerable related software, and can be parsed to separate the software record into its three components. It currently contains 18410 unique vulnerable software records.
OSVDB: This database stores information on whether particular software is affected, not affected or possibly affected, and is thus gives only exhaustive treatment of a software record among these three databases. It can be easily by parsed to separate the software record into its three components and currently contains 9520 unique vulnerable software records. For these reasons, the OSVDB data source was chosen for this information

Vulnerability Discovery, Disclosure and Exploit Release Date Information:

Date information, like software information, cannot be taxonomized, yet is necessary information on a vulnerability. The three major databases, ICAT, BugTraq and OSVDB were considered for this information.

ICAT: Contains a single date field, publish date. However, there is a difference in what this means for dates before and after 2001. For pre-2001 dates this field refers to the date a vulnerability was discovered. For post-2001 dates, this refers to the date a vulnerability was added to ICAT.

BugTraq: Also contains a field indicating the published date. The description of this field is very cursory, stating only that this was, "The date the vulnerability was made public." It does not clearly state if this was the date vulnerabilities are disclosed by any source or if this is the date BugTraq makes its vulnerability record public.

OSVDB: This source contains three potentially useful date fields: Discovery_Date, Disclosure_Date and Exploit_Published_Date. While there are many fields where the particular information is unknown (represented in the database by a date of January 1, 1970) the other fields do yield useful information. These fields are populated by extracting the relevant information from the set of 'external references' listed on the OSVDB site. For these reasons, the OSVDB data source was used to populate CVDB with this information.

8. Findings

The goal of this work was to consolidate the disparate vulnerability data sources so that all information on a particular vulnerability can be queried from a single database. The Exhaustiveness test of CVDB will not only consider the percentage of data was consolidated from the original data source, but also how many records contain the following information: Vulnerability Categories, Dates of vulnerability discovery, disclosure and exploit release, as well as, Severity Information. This is the minimum required information on a vulnerability. Other vulnerability data such as exploit code, though useful, is not necessary for understanding a vulnerability, as it is merely a manifestation of the threat framed by the other vulnerability data.

8.1. Exhaustiveness

8.1.1. Integrated data

This table shows the amount of data that was integrated using this work's consolidation methodology

	Source	Entries	CVDB Entries	Linkage % per Data Source
Category	BugTraq	10906	4136	**37.92%**
Software Details	OSVDB	9520	4997	**52.49%**
Disclosure Date	OSVDB	5335	2064	**38.69%**
Discovery Date	OSVDB	5335	2064	**38.69%**
Exploit Release Date	OSVDB	5335	2064	**38.69%**
Exploit Availability	OSVDB	4358	1702	**39.05%**
Loss Type	ICAT	3797	2905	**76.51%**
Affected Component	ICAT	6692	5129	**76.64%**
Attack Location	ICAT	6694	5138	**76.76%**
Ease of Exploit	CA VIC	8982	3533	**39.33%**
Severity	Secunia	5967	1115	**18.69%**
Solution Status	Secunia	5965	1114	**18.68%**

Table 11: Percentage of Data Integrated into CVDB

A possible reason for the poor integration from various data sources is their level of CVE reference. The greater the reference to CVE, as in the case of ICAT, the greater is the data integrate-able from a chosen data source. However, it must be noted that the criteria for choosing a data source for a particular feature was not that data source's percentage of CVE reference, but rather that data source's taxonomic characteristics for a particular feature. So while greater references to CVE alone, such as in ICAT, gives no indication of a data source's suitability for linking into CVDB, it would, nevertheless, aid in integrating a larger number of vulnerabilities from that source. In addition, the age of the database did not have an influence on the amount of data that could be linked, the oldest database, BugTraq, and the newest database, Secunia, each only integrated 37.92% and 18.69% of their data into CVDB.

There are 413 Full Records which include all of the following information: Category, Relevant Dates and Severity.

Category	4136
Category + Disclosure/Discovery/Exploit Release Dates	1170
Category + Disclosure/Discovery Dates/Exploit Release + Severity	413

Table 12: Number of CVDB Records containing Category, Date and Severity Information

Though this is the bare minimum data required on a vulnerability, records with these three fields populated comprise only 5.48% of the total possible records. This is due to the fact that none of the data sources for Category, Date and Severity information, respectively, BugTraq, OSVDB and Secunia linked well with CVE. Clearly, a greater effort to link to CVE on the part of these data sources is required in order to link a greater amount of data.

8.1.2. Missing Info

Poor solution/action information

There is a critical need for a standardized format solution/action data as it currently exists in free-form text, thus requiring more advanced parsers. However, these parsers may not gather data accurately as there is no agreed upon structure to the solution information. BugTraq's solution information is comprehensive and semi-structured and offers a vast amount of information if laboriously sifted through.

No Linking of Software Products with Known, Vulnerable Libraries

While lists of software which are affected have been integrated, a more fine-grained view of affected targets is required. Often, a vulnerability exists in a shared package or library between multiple applications and at times on even multiple operating systems. A detailed list of libraries and functions called would yield a startlingly pinpointed view of the actual 'location' of a vulnerability. A fix at this level would trickle up and secure the higher level packages as well. The possibility of acquiring this data is highly unlikely with COTS software, though open source developers may be inclined to provide this data.

8.2. Utility to Various Domains

System Administrators

As viruses and other vulnerability exploits increase exponentially in their number and virulence, system administrators are unable to increase the time they spend addressing various holes in their systems at the same rate. Since CVDB can be made CVE-compatible with a modest front-end, system administrators can tie it in with an existing CVE-compatible network application to draw the latest vulnerability information, match new vulnerabilities to applications existing on their network and to receive notifications of critical vulnerabilities in real-time. "CVE-compatible" means that a tool, Web site, database, or service uses CVE names in a way that allows it to cross-link with other repositories that use CVE names. CVE-compatible products and services must meet the four (4) basic requirements in the table below. [CVECOMP]

1.	**CVE Searchable**	A user can search using a CVE name to find related information.
2.	**CVE Output**	Information is presented which includes the related CVE name(s).
3.	**Mapping**	The repository owner has provided a mapping relative to a specific version of CVE, and has made a good faith effort to ensure accuracy of that mapping.
4.	**Documentation**	The organization's standard documentation includes a description of CVE, CVE compatibility, and the details of how its customers can use the CVE-related functionality of its product or service.

Table 13: Requirements for CVE compatibility [CVECOMP]

CVDB can easily meet all four CVE compatibility requirements:

1. CVE Searchable – CVDB allows a user to query for vulnerability information using a CVE ID.
2. CVE Output – In CVDB, it is trivial to include a CVE ID along with other vulnerability information as it uses CVE as its primary key.
3. Mapping – CVDB currently maps to CVE version 20030402, with CAN version 20040802.
4. Documentation – This work serves as a background into how a CVE-based vulnerability database can be used.

Software Developers

CVDB allows developers to easily focus on which vulnerabilities to patch in their current software as well as to design more secure software in the future. Developers can easily query for vulnerable software to retrieve an up-to-date, well-enumerated list of software packages by vendor, product and version number. By ranking these packages by vulnerability severity and overall threat rating or any of the other features available in CVDB, they can prioritize which applications to patch first. In addition, they can query past vulnerabilities for general trends such as the time between vulnerability disclosure and exploit release in order to better 'time' the release of vulnerability information or patches from their side. In addition, with CVDB developers have access to a full set of information on the potential sources of attack, trends in the vulnerabilities exploited, general classes of vulnerabilities they need to correct in their software and vulnerable system components and software libraries. This information would be invaluable to them in nearly every phase of a software development lifecycle, whether it be the requirements gathering/definition, High-Level Design, Implementation or Testing Phases.

In addition, developers can use distance measures to place new vulnerabilities in existing categories. Thus whenever a new bug is found, developers can leverage the large volume of vulnerability data present on possible attack vectors and remedies/solutions for previous *similar* bugs, enabling new *bug* fixes and patches to be rolled out quicker and in a more streamlined fashion.

Software Vulnerability Researchers

CVDB allows vulnerability researchers to query every one of the features listed in Section 7.3 as can any software developer or system administrator. They can also use distance measures to find *similar* vulnerabilities. However, while the other two user groups would be content with just the information

pertaining to their network or their specific product line, vulnerability researchers would need to understand the *relationship* between the data, in order to draw conclusions on overall trends in the industry across vendors.

Data mining algorithms can discover existing, yet hidden, patterns in data as well as predict to which group(s) a particular new vulnerability belongs. These algorithms require a period of *training* before they can be expected to yield reasonable predictions and require a large set of data for this [SMHB]. It is argued that the size of the vulnerability data and the numerous features available in CVDB are sufficient to enable such research to flourish. Applying data mining techniques, CVDB contains a large consolidated set of data with which researchers can study such questions as: [LGH96]

- With what frequency and in which components do certain classes of vulnerabilities occur?
- What criteria are important in predicting the release and severity of the next exploit?
- What commonalities exist in vulnerabilities between operating systems?

These represent a tiny fraction of the possible insights waiting to be gleaned from CVDB. The task of finding those is left as an open work for a future researcher.

8.3. Extensibility

The ability for the current work to grow is predicated on two factors: new information that is not represented already and CVE IDs with which to link that new vulnerability data to the current schema. While it may be tempting to use other IDs available in the database such as BugTraq or OSVDB IDs, this is highly discouraged. Though some additional data may be accessible by checking what other sources reference Non-CVE IDs, then checking to see if that ID is currently referenced by CVE, the use of non-primary keys for referencing data would lead to the problems associated with transitive dependencies. This would go against the principle of this work, to see how much information could be consolidated in a structured manner. CVE has strong support in the vulnerability database industry: BugTraq references CVE in a unique field in its database, the CoopVDB & ICAT databases are entirely built around CVE, and another vulnerability database provider, ISS, is a founding member of CVE. Therefore, it is argued that not only will currently available data become easier to integrate into CVDB in the future, but that data on newer vulnerability aspects released in the future data will reference CVE by default.

8.4. Scalability

The scalability of this work will be evaluated on how quickly it integrates vulnerability information from the time they are disclosed on any of the five data sources. The sole determining factor in this is how quickly vulnerabilities on those data sources reference CVE IDs; once they reference a CVE ID, integration into CVDB is a trivial matter.

All five data sources contain a published date and reference CVE ID for their vulnerabilities. We will, thus, be able to surmise how many years need to pass to achieve a certain percentage of "CVE saturation" in a particular year's set of vulnerabilities. We will now look at each data source and see how quickly it

adds a CVE ID. In the tables below, the * in the 2004 columns indicates that the values have been pro-rated until the end of 2004.

8.4.1. CVE Adoption by Major Databases

BugTraq

Years Back	5	4	3	2	1	0
Year	1999	2000	2001	2002	2003	2004*
Vulns with CVE IDs	411	938	889	1278	512	521*
Total Vulns	627	1158	1457	2583	2644	2517*
%	**65.6%**	**81.0%**	**61.0%**	**49.5%**	**19.4%**	**20.7%**

Table 14: CVE Adoption by BugTraq

The table above shows the percentage of "CVE Saturation" among Bugtraq's vulnerabilities in a year, going back to 1999, the year CVE began. It shows that 20% of BugTraq's vulnerabilities released in the last year are linkable. This trend grows to 50% of vulnerabilities if we go back 2 years, to 60% for 3 years, and up to 81% if we go back 4 years.

OSVDB

Years Back	5	4	3	2	1	0
Year	1999	2000	2001	2002	2003	2004*
Vulns with CVE IDs				153	605	1702*
Total Vulns				181	1100	4863*
%	**n/a**	**n/a**	**n/a**	**84.53%**	**55.00%**	**41.99%**

Table 15: CVE Adoption by OSVDB

The table above shows the percentage of "CVE Saturation" among OSVDB's vulnerabilities in a year, going back only to 2002, the year OSVDB began publishing data. It again shows a similar trend to the BugTraq data.

ICAT

Years Back	**5**	**4**	**3**	**2**	**1**	**0**
Year	**1999**	**2000**	**2001**	**2002**	**2003**	**2004***
Vulns with CVE IDs	1499	1202	1360	1494	937	620*
Total Vulns	1551	1203	1395	1573	1050	1228*
%	**96.65%**	**99.92%**	**97.49%**	**94.98%**	**89.24%**	**50.51%**

Table 16: CVE Adoption by ICAT

The table above shows the percentage of "CVE Saturation" among ICAT's vulnerabilities in a year, going back to 1999. As ICAT uses CVE IDs as its primary key, the CVE reference is the primary key itself, therefore, unlike the previous tables, this table show the percentage of ICAT IDs of a particular CVE year

that exist out of the total CVE ID pool for that particular year. In fact, even this data shows a similar trend to that of the other two major databases, OSVDB and BugTrag.

8.4.2. CVE Adoption by Minor Databases

Secunia

Years Back	5	4	3	2	1	0
Year	**1999**	**2000**	**2001**	**2002**	**2003**	**2004***
Vulns with CVE IDs				35	1178	1576*
Total Vulns				728	2716	3026*
%	**n/a**	**n/a**	**n/a**	**4.81%**	**43.37%**	**52.08%**

Table 17: CVE Adoption by Secunia

The table above shows the percentage of "CVE Saturation" among Secunia's vulnerabilities in a year. We only go back to 2002, the year Secunia began publishing data. It shows that 52.04% of Secunia's vulnerabilities released in the last year are linkable. This trend actually decreases to 43.37% of vulnerabilities if we go back to 2003 and to near zero (4.81%) if we go back to 2002. This would most likely be a result of Secunia publishing a large number of vulnerabilities that already have had CVE IDs assigned in previous years, and is not a reflection on future linkability of Secunia IDs.

Computer Associates' VIC

Years Back	5	4	3	2	1	0
Year	**1999**	**2000**	**2001**	**2002**	**2003**	**2004***
Vulns with CVE IDs		1080	520	596	651	836*
Total Vulns		2639	880	1288	2385	2056*
%	**n/a**	**40.92%**	**59.09%**	**46.27%**	**27.30%**	**40.63%**

Table 18: CVE Adoption by Computer Associates' Vulnerability Information Center

The table above shows the percentage of "CVE Saturation" among CA VIC vulnerabilities in a year. We only go back to 2000, the year CA began publishing data. No clear trend is discernable from this data.

8.4.3. CVDB Scalability Discussion

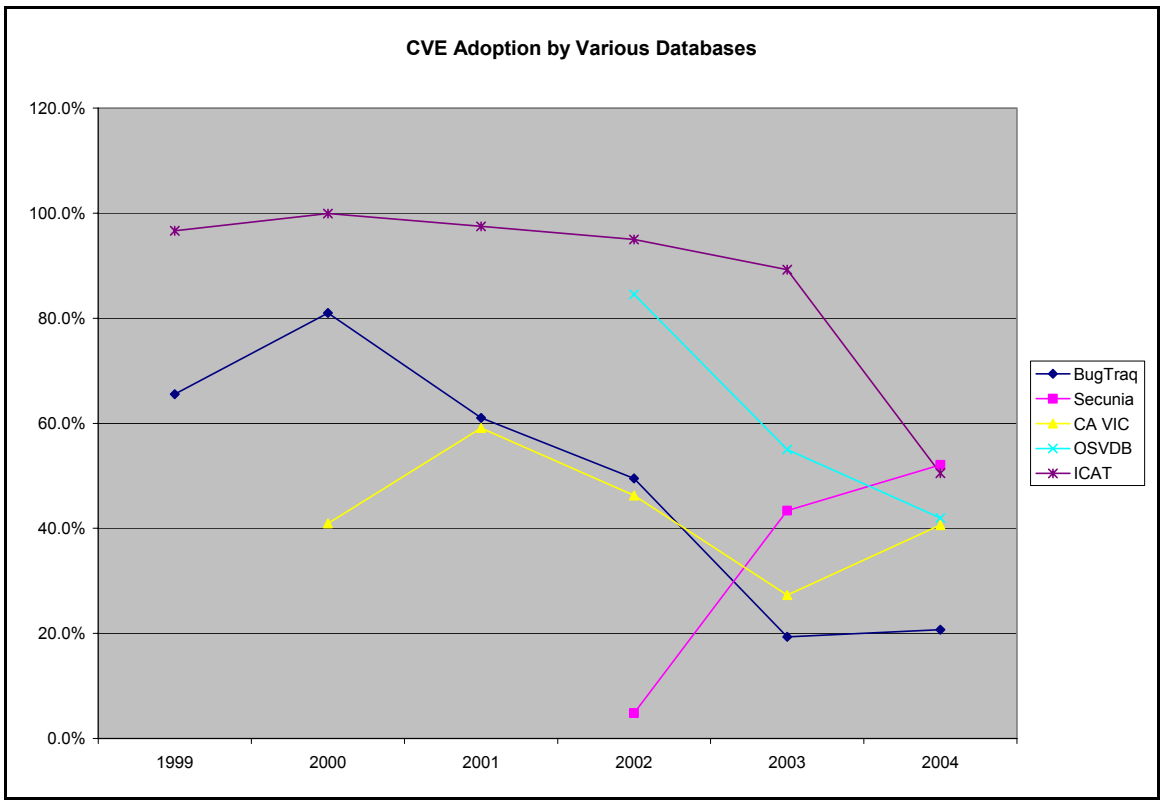

Chart 9: CVE Adoption by Various Databases

Though there is still not enough conclusive data to make a long-term forecast of how well CVEs will be adopted by databases in the future, it is readily apparent from the major database data, that CVDB is incapable of linking the majority of recent vulnerabilities. This chart seems to indicate that any data source is only able to link to approximately 40% of its current year's data to CVE. If it can be shown that many exploits compromise recently disclosed and discovered vulnerabilities, then this has grave implications for the utility of CVDB for use in automated network tools and for day-to-day security practitioners. This can be countered by the argument that any critical vulnerability would likely have both a CVE ID assigned and be referenced by the major vulnerability databases. We will conduct a preliminary survey of two vulnerability data sources, which list top or severe Vulnerabilities, SANS and CERT.

Sans.org maintains a list of Top 10 vulnerabilities each for Windows and UNIX. These are grouped based on type of application, such as Database Server, Mail Server, Web Server. Within each of these groups, specific applications are enumerated. A snapshot of the Top 20 SANS vulnerabilities for November 19, 2004 reveals that of the 20 groups of current Top vulnerabilities, 19 groups refer to CVE, a ratio of 95%.

A more detailed study of SANS's Top 20 vulnerability [SANST20] list is perhaps beyond the scope of this paper.

Next, we will take a cursory look at CERT's vulnerability notes [CERTVN]. CERT assigns a severity metric to vulnerabilities from a scale of 0 to 180, and releases advisories for vulnerabilities receiving score of 40 or more. In 2004, CERT has assigned a severity metric of 40+ to 11 vulnerabilities, of which 8 vulnerabilities refer to CVE, a ratio of 72.7%.

Perfunctory investigation of SANS and CERT suggests:
CVE IDs *are*, in fact, assigned for even recent vulnerabilities.
Major vulnerability databases are successful in linking to CVE for the severe vulnerabilities, ones that are likely to cause the most harm on a network.

So, if further in-depth analysis can show that the majority of those vulnerabilities which currently have exploits are also linked to by the major databases, then we may not need to sound the death knell for CVDB's utility in automated network vulnerability tools.

9. Future Work

9.1. Canonicalization of Additional Vulnerability Information

Data from disparate sources such as vulnerability titles, descriptions and solution information, hold valuable information, yet are in different formats. For example, the vulnerability title for a buffer overflow in Sun's rpc daemon has the following titles in various data sources:

CVE	Buffer overflow in NIS+, in Sun's rpc.nisd program
ICAT	Buffer overflow in NIS+, in Sun's rpc.nisd program
BugTraq	Multiple Vendor NIS+ Buffer Overflow Vulnerability
CA VIC	Solaris rpc.nisd buffer overflow vulnerability

Table 19: Vulnerability Titles in Various Data Sources

They each convey the same idea easily for a human reader, yet, differences exist among them. Likewise, certain OSVDB records contain solutions for their respective vulnerability, yet they are not in a common or standardized format.

Canonicalizing this data requires, first, that templates are agreed upon to store desired information in a vulnerability title, description or solution text. Then, a decision tree must be created to unambiguously populate these templates from the existing data. At present, this would be most accurately accomplished manually. Valuable information can, however, be gleaned from these fields by applying an intelligent data mining tool. This tool needs to know both the syntactic and a semantic meaning of the information in order to accurately parse it, simply for keywords. Once keyword parsing is complete, the templates would be automatically populated based on more intelligent decision trees in order to feed it into a database.

9.2. Using Non-CVE IDs as Primary Key

Though BugTraq and OSVDB IDs were not considered for primary keys based on the criteria in Section 0, they would be excellent candidates for study into consolidated database creation. Certain of the selections for feature data sources would change in favor of an OSVDB or BugTraq-based source. Other features would be nearly fully populated such as vulnerability categories if BugTraq-based primary keys or exploit availability if OSVDB-based primary keys were used. The use of ICAT however would cause no change in the amount of data linked as ICAT uses CVE as its primary key. In fact, as ICAT's addition of new CVEs to its database lags behind by a week or so, often, the most current data would not be available to an ICAT ID-based database.

9.3. Open Issues

Current data sources are missing causal information on the pre-conditions which vulnerabilities require to be exploited. While basic location of exploit information is available, via OSVDB and other sources, data about the conditions, such as requiring a denial of service for a vulnerability to be exploitable is still not available.

There are still many features of vulnerabilities which are not detailed in the current data sources, for example, possible "protection barriers" or "mitigation factors" which are the components which may protect or isolate assets from particular attacks. [HOLL]

A further extension to this work would be models which prove that the data can be easily accessed by Intrusion Prevention Systems to take remedial actions such as blocking firewall ports or installing the latest patches.

10. Related Work

As merely storing data on a vulnerability serves no purpose, other researchers have sought to work out ways to describe and model attacks on computers and networks in the hopes of finding a top-down solution to the computer and networks security situation. Work in this field was done by John Howard and Tom Longstaff [HL98] with the goal of creating a "common language" for exchange of information between many security related components.

Another type of attack model includes attack graphs [JSW], which aim to map out all the potential scenarios by which a vulnerability may be exploited. These graphs can then be used to find the minimum number of components which need to be secured in order to prevent an attack. Further uses of attack graphs include the ability to suggest which attacks would be the most cost-effective to protect against.

Another recent work on 'providing a common language for security incidents' is OVAL, the Open Vulnerability Assessment Language, by Mitre. Its stated mission is to become *the* language to determine the presence of software vulnerabilities. While its schema does address many features normally found in a vulnerability database, it is not well suited for a taxonomical approach to organizing vulnerability data. However, if OVAL's promise holds, it may in address the needs of system administrators to receive network vulnerability assessment and remediation data in a structured format. OVAL does not yet promise to automatically fix vulnerabilities that are found.

11. Conclusion

Based on the work done in this paper, there is strong evidence that a consolidated taxonomy encompassing and organizing all relevant data can be achieved. This current work, however, is far from that goal. Three primary obstacles exist: lack of referencing a common 'primary key', available, though un-structured and free-form descriptions of necessary vulnerability data, and lastly, lack of data on all aspects of a vulnerability. The evaluation of existing primary key candidates suggests that Mitre's CVE ID can be *that* link between all existing vulnerability data as it is open, supported by major security firms, contributed to by major security practitioners, and already *the* primary external reference for other vulnerability databases. In addition, the information that is currently available, on solutions to vulnerabilities, for example, is mostly free-form requiring intelligent parsing by a human or sophisticated, data mining algorithm. The greatest complication to overcome, however, is the lack of comprehensive vulnerability data, key sources which include victims of security incidents, vendors of faulty equipment and the research community. It is surmised that the lack of valuable vulnerability data can be attributed to the reluctance of the victims of security incidents to share their data, the unwillingness of proprietary software vendors to release certain implementation-level details, such as libraries used, as well as their indisposition to allocating greater resources to vulnerability testing in their products. Therefore, the best hope, currently, for advances in the state of available vulnerability data lies in the research community.

This work has focused on one manner of integrating all currently available information. It has only considered data that can be unambiguously extracted from various data sources by simple parsers. It is felt that even with the use of more advanced, information mining tools, which can wade through the sea of unstructured vulnerability data, this current integration methodology would still provide repeatable, unambiguous, and exhaustive results. Though the goal of coalescing all available data, which would be of use to automated tools, security practitioners, and vulnerability researchers, is not yet achieved this work has resulted in the most exhaustive collection of vulnerability data to date.

12. Appendix I: Unused Data Sources

12.1. US-CERT

Though the US-CERT Vulnerability Notes Database [CERTDB] contains useful information on a vulnerability, including additional features such a severity metric, this database was not highly referenced by other sources. There are only 245 references by CVE, 381 references by CA VIC, 0 by Secunia, 302 by BugTraq and an equal or lesser number of references by ICAT, as ICAT uses the same references as CVE. A likely reason for this is the conservative approach taken by CERT in releasing vulnerability information to the public. CERT's full disclosure guidelines state that they will release vulnerability information "45 days after its initial report, regardless of the existence or availability of workarounds", unless "extenuating circumstances require an earlier or later disclosure." [CERTFD] The reality tends towards a later disclosure. It must be noted that CERT maintains an internal list of vulnerabilities, even prior to public disclosure, but these entries were not available for this database. The effect of CERT's goal and perhaps, a duty, to act in the "best interests of the community overall" is a public database updated much later than others.

12.2. ISS X-Force

The Internet Security Systems' X-Force database, publishing vulnerability information since 1994, is one of the oldest repositories of vulnerability information. ISS is one of the founding members of CVE, and is referred to by CVE 4920 times.

Yet, it was not evaluated for this taxonomy for a variety of reasons. One is its proprietary control; X-Force manages the structure and entries in the database. As X-Force is also a vendor of software products, the potential for reporting discrepancies may arise wherein X-Force may release competitor's information early, without proper checking, and be slower to add vulnerabilities in its own products claiming they are verifying the accuracy of the vulnerability claim. While no proof is available this is the case, the X-Force FAQ [XFFAQ] clearly allows for potentially adding inaccurate information. In Section 2.6, it admits that X-Force will add a third-party product's vulnerability data "based on the credibility of the source reporting the issue." It further states that it will only remove such an entry if the "non-existence of the security issue" is reported by a credible source. Therefore, a great deal of ambiguity regarding what constitutes a "credible source" exists, and is solely determined by the needs of X-Force.

However, the X-Force database's propriety nature and ambiguity in sources are not the primary reason it was not evaluated for this taxonomy; rather it was due to the lack of unique and linkable information not represented elsewhere. Of the 4920 references by CVE, only 141 entries where not referenced by BugTraq. Therefore, to have included X-Force would have only added redundant data to the database.

Furthermore, the only possible value-add the X-Force database represents is in terms of its Risk Level Ratings. Yet, from X-Force FAQ's description, the criteria for choosing between the three risk levels is ambiguous and arbitrary.

High	Security issues that allow immediate remote or local access, or immediate execution of code or commands, with unauthorized privileges. Examples are most buffer overflows, backdoors, default or no password, and bypassing security on firewalls or other network components.
Medium	Security issues that have the potential of granting access or allowing code execution by means of complex or lengthy exploit procedures, or low risk issues applied to major Internet components. Examples are cross-site scripting, man-in-the-middle attacks, SQL injection, denial of service of major applications, and denial of service resulting in system information disclosure (such as core files).
Low	Security issues that deny service or provide non-system information that could be used to formulate structured attacks on a target, but not directly gain unauthorized access. Examples are brute force attacks, non-system information disclosure (configurations, paths, etc.), and denial of service attacks.

Table 20: ISS X-Force Risk Level Criteria from X-Force FAQ [XFFAQ]

12.3. The Public Cooperative Vulnerability Database at CERIAS

As this database was the off-shoot of work done by Ivan Krusl in [Krusl98], the last major published work on taxonomy consolidation, and by others at the CERIAS center at Purdue, it was hoped that this would form a valuable reference in the current work. However, based on the information on the site and the fact that no new papers on the subject have been published by the center since 1998, it seems to be an abandoned project. The Public Cooperative Vulnerability Database [COOVDB] uses the CVE ID as its primary key similar to the ICAT database, yet, contains no additional information over and above ICAT. In fact, it seems not to have been updated in a while; many "reserved" CVE IDs from 2002 are still in "reserved" state in this database, while CVE and ICAT have full information on them.

References

[AS95] Aslam, T. 1995. A Taxonomy of Security Faults in the Unix Operating System. M.S. thesis, Purdue University.

[AS96] Aslam, T., Krsul, I., and Spafford, E. 1996. Use of A Taxonomy of Security Faults. In 19th National Information Systems Security Conference Proceedings. Baltimore, Maryland.

[BB96] Bishop, M. and Bailey, D. 1996. A Critical Analysis of Vulnerability Taxonomies. Technical Report. CSE-96-11, Department of Computer Science at the University of California at Davis. September.

[CERTDB] US-CERT Vulnerability Notes Database. http://www.kb.cert.org/vuls/html/search, 28 Nov 2004.

[CERTFD] CERT/CC Vulnerability Disclosure Policy. http://www.cert.org/kb/vul_disclosure.html, 28 Nov 2004.

[COOPVDB] CERIAS Co-Operative Public Vulnerability Database. https://cirdb.cerias.purdue.edu/coopvdb/public/, 28 Nov 2004.

[CVECOMP] CVE-Compatible Products and Services. http://cve.mitre.org/compatible/, 28 Nov 2004.

[HL98] Howard, John D. and Longstaff, Thomas A. 1998. A Common Language for Computer Security Incidents. Sandia National Laboratories, Report: SAND98-8667.

[HOLL] Hollingworth, Dennis. Towards Threat, Attack, and Vulnerability Taxonomies. Network Associates Labs.

[JSW] Jha, Somesh, Sheyner, Oleg and Wing, Jeannette. 2002. Minimization and reliability analysis of attack graphs. Technical Report CMU-CS-02-109, Carnegie Mellon University, February.

[L93] Landwher, C., Bull, A., McDermott, J., and Choi, W. 1993. A Taxonomy of Computer Program Security Flaws. Tech. Rep. NRL/FR/5542{93-9591, Naval Research Laboratory, November.

[LGH96] Landwher, C., Bruce C. Gabrielson, and Humphrey, Jeff. 1996. Requirements and Approaches for a Computer Vulnerability Data Archive. Invitational Workshop on Computer Vulnerability Data Sharing, Gaithersburg, MD, June 10-12.

[OSVAIM] OSVDB: Project Aims. http://www.osvdb.org/OSVDB-Aims.php, 28 Nov 2004.

[PA] Carlstead, J., Bibsey II, R., and Popek, G. 1975. Pattern-Directed Protection Evaluation. Technical Report, Information Sciences Institute, University of Southern California. June.

[RISOS] Abbott, R. P. et al. 1976. Security Analysis and Enhancements of Computer Operating Systems. Technical Report, NBSIR 76-1041, Institute for Computer Science and Technology, National Bureau of Standards.

[SMHB] Schumacher M., Haul C., Hurler, M. and Buchanan, A. 2000. Data Mining in Vulnerability Databases. Department of Computer Science, Darmstadt University of Techonology, 22 March.

[TAL97] Longstaff, T. 1997. Update: CERT/CC Vulnerability Knowledgebase. Technical presentation at a DARPA workshop in Savannah, Georgia.

[XFFAQ] X-Force FAQ. http://xforce.iss.net/xforce/xfaq/, 28 Nov 2004.