Final Report of the 2nd Workshop on Research with Security Vulnerability Databases, January 1999

Pascal C. Meunier and Eugene H. Spafford
CERIAS
Purdue University
West Lafayette, IN 47907
pmeunier@purdue.edu
spaf@cs.purdue.edu

CERIAS Technical Report 99/06

Abstract

This report presents the results of the workshop and the reports of the working groups on sharing vulnerability data. We divided the obstacles to sharing vulnerability data into technical, motivational and consequential issues. We summarize and expand on the discussions of fundamental issues such as nomenclature, vocabulary, and the contents of vulnerability databases. There was considerable goodwill expressed at the workshop in favor of sharing vulnerability data. We discuss the respective advantages of the Open, Centralized, Federated, and Balkan data sharing models and the motivational obstacles to such sharing. We observe that no single proposition satisfies all parties involved; therefore, the parallel pursuit of two that cover the maximum number of interested parties, with minimal overlap, has the best chances to achieve success.

Table of contents

Introduction

As for all databases, vulnerability databases provide useful reference tools. They are used for education (research), vulnerability scanning, intrusion detection, software engineering, and computer forensics. However, the construction of a vulnerability database is much more complex than archiving typical account information: the data that is to be stored often needs to be generated and verified from rumors or incomplete reports, sometimes by unknown or untrusted people. Moreover, research databases don't simply archive the raw information, but extract relevant features of the vulnerabilities. The analysis of vulnerabilities is complex enough to be the subject of Ph.D. theses (2); yet it is the results of the analysis of each vulnerability that are to be stored in the database. Therefore, the schema (structure) of the database depends on how the vulnerabilities are analyzed.

The first workshop on Research with Security Vulnerability Databases was sponsored by NIST in 1996. While the great majority of people thought it was a good idea to share vulnerability data, in practice the sharing encountered many obstacles. Since the workshop, many different vulnerability databases have been produced, but there are no publicly known cases of database sharing. The obstacles to data sharing are 3-fold: technical, motivational and consequential.

Technical obstacles include concerns about the usefulness and correctness of the data, the format of the data, the identification of the vulnerability, the analysis performed, and even the language used to communicate the information. However, even if those were overcome, motivational obstacles remain. The construction of the databases is an expensive process, and the exchange needs to be fair. In addition, the ignorance of others is to the advantage of the one who detains the information. Thus, the proverbial "tragedy of the commons" situation applies because the general interest for sharing the data is at odds with individual interests.

Finally, the consequences of sharing the data are potentially considerable, because the information could be used to attack valuable targets. For this reason the U.S. government has been hesitant to authorize the publication of vulnerability information obtained with public funds. In addition, there is the possibility of lawsuits against the owner of the database by victims of attacks, or by an entity owning a product said to contain a vulnerability.

The goal of the second Workshop on Research with Security Vulnerability Databases was to discuss the following topics:

How to organize common schema;
How to collect "good" data;
How to exchange database information with trusted parties;
Standards and conventions for presentation and naming;
Using these databases in research;
Whether such databases pose a threat and how to limit such a threat;
How to maintain currency of such databases.

As part of achieving these goals, the second Workshop on Research with Security Vulnerability Databases investigated the mechanisms, pitfalls, and motivations for sharing, or not, vulnerability data. The attendees were divided into 5 groups; the first 4 groups were each to discuss a model for data sharing, whereas the 5th group discussed fundamental issues such as nomenclature and definitions. The first 4 groups discussed (in order of assignment) the “Fully Available”, “Centralized Model”, the “Federated Model” and the “Balkan/Status Quo” models. The original reports are in their respective appendixes at the end of this document. We edited and added to the various reports from the 5th group and discuss the aftermath of the workshop.

Part I: Technical issues

Language

The sharing of information is difficult without a common language. The focus on information security is even more recent than computer science itself, the available vocabulary to discuss computer security concepts is limited. This leads to an “overloading” of terms, i.e., a reuse of the same terms with varying scope and level of abstraction. This was observed to lead to confusion and misunderstandings among workshop participants.

The identity (names) of the vulnerabilities discussed must be unambiguous. Two solutions were proposed: a Common Vulnerability Enumeration basically consisting of a number assigned to each vulnerability, and a naming convention. The naming convention was envisioned as a principled naming scheme that would be consistent and repeatable. The name itself would indicate the nature of the vulnerability. The following requirements for a naming convention were identified:

1) It should avoid reliance on taxonomic information and focus on easily observable features, e.g., names of the application, service,

protocol, and/or vendor, etc.

2) It should have the basic characteristics of good taxonomies, e.g. repeatability, specificity, exhaustiveness, and mutual exclusiveness.

3) When deriving the name, the exploit should be separated from the inherent vulnerability where possible.

4) Aliases for the standard name should be supported, e.g. to recognize "well-known" names for vulnerabilities such as Smurf or Ping of Death.

Built into the requirements was the problem that the design and application of the naming convention required an understanding and analysis of the vulnerability prior to naming it. Therefore, naming new vulnerabilities could be difficult, and the name could change if the understanding changed. Moreover, the design of the naming convention with the above characteristics was a considerable challenge. This highlighted the practical advantages of a Common Vulnerability Enumeration (1). Consequently, MITRE was encouraged to work on the CVE after the workshop. The CVE has been described in the proceedings of this workshop (1), at several conferences in 1999, and was made public in the fall of 1999 (http://cve.mitre.org).

Vocabulary issues

The following vocabulary issues were discussed by group 5a:

> a. In a specific application, instance or use of a product: a vulnerability is a feature that allows harmful, unintended behavior. In a product prior to use: a [potential] vulnerability is a feature that may be a vulnerability in some intended application, instance, or use. Notice that the specification of the context is necessary because not all consumers of a product will have the same intention or the same understanding of harm. For example, a video screen that can be seen by anyone may be an asset in an airport as well as a liability [i.e. vulnerability] in a classified environment. A vendor who wishes to sell video screens to both communities may need to regard ease of viewing as a potential vulnerability.

b. The above discussion of vulnerabilities manages to avoid the word "policy" (although in practice the context may imply the policies). In some cases, intended, harmless behavior may be obvious, in others it may be codified by policy documents. The Common Criteria (CC; http://cctoolbox.sparta.com/) allows for both situations.

In those cases which are policy based, there is the immediate question of what "policy" means. One possibility is that of an organizational security policy. The other is that of a technical security policy. The latter assumes more layers of prior analysis. Typically, a technical policy is invented in response to a corresponding organizational policy.

These two kinds of policy are sufficiently different as to cause misunderstandings whenever they are confused. In fact, this was one of the first confusions to be unraveled in the construction of the CC and its predecessors.

c. Atomicity of incidents and attacks. For some, an attack is an atomic event while, for others, it is a sequence, scenario, or pattern of events. The same is true of the word incident. Howard's thesis (2) illustrates both perspectives by treating an incident as a sequence of attacks.

d. Abstraction level of attacks. We found it useful to consider at least three levels of abstraction:

- The concrete level: an attack instance [or incident] at a particular time and place. Concrete attack instances are what get reported to CERT (although some sanitization may occur as part of the reporting process).

- The "portable" level: a sanitized attack is the set of all repetitions of a concrete attack instance. Information that is not of general interest is filtered out, e.g., time and location of the attack, details about the extent of tangible loss, and perhaps details of the system configuration under attack as well. Although we did not come up with a precise definition of "all repetitions," we believe that this level is well understood in practice.

- The reusable-information level: an abstract attack consists of the common security-relevant characteristics of a set of similar attacks. An abstract attack is normally platform

independent. We believe that the community does not yet have a general ability to present attacks at this level because the community hasn't yet converged on the salient, reusable attributes of concrete attacks.

e. Response to attack. The Common Criteria (CC) and the "MITS" Report (6) use the terms "countermeasure" and "safeguard," respectively. Interestingly, these two words have different meanings in different communities. To some military folk (but not in the CC), a countermeasure is considered to be a preventive measure, whereas a safeguard always includes the broader alternatives of attack identification and attack recovery. To many civilians, the term countermeasure has a hostile connotation and is thought to be analogous to counterattack.

f. The word "system" has at least two good uses in connection with attacks: Sometimes the system is the target of discussion. The CC uses the term Target of Evaluation (TOE) in preference to system in this sense. Sometimes the [whole] system is the big picture that contains the target of discussion as well as its users and relevant aspects of the target's environment. The CC uses the word environment to refer to the whole big-picture system, something that takes a little getting used to for some people.

g. People use various terms with multiple overlapping meanings for what an attack might compromise. The root words in these terms are resource, asset, and information. For some people, assets are normally monetary in nature. Most people agree that information is a type of resource. However, the CC defines assets as information and resources. For the sake of discussion, let us take resource to be the general term that includes information and monetary assets, among other things. What resources are relevant in discussions of attacks and vulnerabilities? This question tends to generate a surprising amount of heat. Without answering the question, let’s at least look at some categories of resources:

IT resources (e.g., information, processing capabilities)
Non-IT resources (e.g., money, property, life, liberty)
Target resources (part of the target of discussion)
Environmental resources (outside the target)

In conversation, people often equate resources with IT resources or with target resources. All but one of our group agreed that all kinds of resources are of potential interest, at least at the level of concrete attacks. A telling example was a device that talked with satellites. One could issue a sequence of commands that caused the satellite to go off line, at which point the cheapest recovery strategy was to launch another satellite -- at a cost of about $22 million. In this concrete example, the monetary data provides a clear understanding of the attack's significance. In the context of a more abstract attack description, the main point might be simply that attacks that necessitate remote maintenance are good candidates for preventive safeguards.

While the above observations do not provide definite answers to how each term should be used, being aware of the different possible interpretations will help communication. Whereas some other vocabulary problems may be encountered, they suggested filling deficiencies by "stealing" terminology from other sources when feasible. Other good sources include the Common Criteria (CC) and the "MITS" Report (ISO/IEC TR 13335-1), the preceding and first workshop, as well as interested professional groups, e.g., security auditors and security consultants.

Contents of Vulnerability Databases (VDBs)

The content of vulnerability databases is under debate, because attacks might cause damage in the wrong hands, and because different things are of interest to different people. The conclusion reached by group 5 was that attacks, incidents, tools, patches, system component identifications, system and organizational policies, asset valuations, and probabilities of exploit should all be included. The reasons given were that:

a) All of this information is useful, and it should be included even if some items are of interest only to some users.

b) Including related concepts in a VDB allows richer linkage and thus better utility. For example, one may want to query on attacks to find related vulnerabilities of interest (or *vice-versa*).

c) Because of apparent diversity of opinion on the content and

structure of VDBs it is important for them to be empirically grounded. That is, they should be grounded directly or indirectly on raw incident data, including any information that may be needed to assess seriousness (e.g., tangible loss, frequency of exploit, available patches, identification tools).

A tentative list of useful items in a vulnerability database was proposed by Dale Blommendahl, which we extensively edited (see below); however, some of these are not always available, especially if one attempts to construct a database from public records:

1. Templates to define the information system involved
    a. System components
        1) Storage
        2) Sommunications interfaces
            a) Protocols & services
        3) Application processes
        4) Operating system processes
        5) Security mechanisms
        6) Userids
        7) Terminals
    b. Communications mediums
        1) Direct copper wire
        2) Direct fiber
        3) Shared copper wire
            a) Ethernet
            b) Telco network
            c) Token ring
        4) Shared fiber
    c. Configurations
        1) Permissions
        2) Privilieges (e.g., setuid bit)
        3) Entities
        4) Filters
        5) Etc...

2. Policies
3. Detection methods
    1) Attacks
    2) Telltale signs
    3) Version numbers
4. Countermeasures
    1) Controls

2) Patches or code revisions
3) Mitigating configurations & environment

However, the danger is to transform the vulnerability database into an incident database. It is worth noting that much information in current databases actually relates to attacks, rather than to the actual underlying vulnerability. For example, relevant security policy(ies) are specified in relation to the possible attacks. Because attacks and vulnerabilities may have many-to-many relationships, attacks add an extra level of indirection between policies and vulnerabilities. Therefore, policies are only indirectly related to vulnerabilities.

The fundamental problem is that vulnerabilities do not exist in a vacuum, but are related in a complex manner to a number of other "objects": hardware, software, patches, attack scripts, policies, assumptions, threatened or exploited resources, incidents (instantiations of possible attacks), etc. It is those relationships that define the characteristics of the vulnerability and the usefulness of the database. The database is not just a collection of facts, but a model representation of what a vulnerability is and does.

Some of these objects in the database are the same, but with different values. One of the current challenges for vulnerability databases is the ability to present the data at different levels of abstraction, and from different perspectives (e.g., policies, attacks, vulnerabilities, target resources, software). The current weakness of vulnerability databases, in our opinion, is the inability to show which objects are similar, to abstract the encompassing class of these objects, and to "pull" on any one of those objects and present the database from that point of view. A built-in abstraction mechanism would be the definition of all the objects as instances of classes, with the classes themselves forming a structure between levels of abstraction. This is why an object-oriented database is more likely to be a satisfying and valid representation of vulnerabilities, although it would be much more difficult to implement than a collection of attacks.

Data sharing will remain difficult (expensive) as long as there is no agreement on what is relevant vulnerability data (4). The participants of all groups agreed on the need to define a common data exchange schema. Moreover, a point was made that the taxonomy must be established first, as it will guide the rest (5). However, inasmuch as the database is a model of the reality (as discussed above), the taxonomy and what is considered the relevant data will vary and evolve. Group 1 emphasized a

flexible XML format. However, transferring data between traditional databases and object-oriented ones may prove an additional challenge.

Part II: Motivational issues

The benefits of sharing vulnerability information are better vulnerability scanning tools, intrusion detection software, education, research, the formulation of security standards and objectives, avoiding the duplication of efforts (lower economic cost) and of course more secure software. However, the obstacles to such sharing range from basic self-interest, the lack of mechanisms to share fairly, the fear of liabilities, ethical motives, and the risk of losing revenues by acknowledging vulnerabilities in one's own software.

An important realization obtained from the workshop was that no matter what the technical issues are, there will always remain entities choosing to keep information private for a perceived advantage. Trying to find the single perfect solution, or requiring or expecting complete cooperation from everyone is unrealistic. The report by group #4 (appendix D) is a fair account of the motivational obstacles to data sharing, with possible mitigators. It seems that the best solution in order to achieve results in a reasonable time frame is to push ahead with one or two models, with the awareness that neither is a perfect solution and neither will be embraced completely.

From the reports of the working groups and personal conversations, the proposals of the fully available model (group 1, Appendix A) and the centralized model (group 2, Appendix B) converged and had significant areas of overlap. Therefore, the people interested in either model A or B could be served adequately by just one with a few compromises or by a merger of the two models. The resulting model would serve best micro-companies, education, research and individuals because of the low overhead. This database should contain all the vulnerabilities enumerated in the MITRE CVE. A possible pitfall of this model, like all volunteer efforts, is the possible degeneration of the database into a moribund state because of the lack of committed and motivated (paid?) leaders and editors.

The proposal for the federated model is significantly different (Appendix C). It allows contributors (members) to retain control of their database, while still achieving sharing. It may be viewed as a safer investment from a corporate perspective and requires more significant

resources from the participants, and so would most likely serve corporations better. The people interested in this model most likely would not be served adequately only by the proposals of groups 1 and 2. A possible pitfall of this model is that the independence and power of the participants is such that they might not stay fully committed to the effort, and politics may play a role in the workings of the database.

Whereas no single proposition satisfies all parties involved, the judicious, parallel pursuit of two has the best chances of satisfying the needs of many. Because of the small overlap between the two models, the parallel pursuit of both instead of a single one will not dilute efforts considerably. Moreover, if one should fail, the other one may still achieve significant benefits for the participants.

Part III: Consequential issues

Since this type of dangerous data sharing is a new experience, participants expressed caution. The issues are the legal consequences in the US or abroad, and also the consequences for the security of the individuals, companies and nations using vulnerable software. It should be noted here that the proposal by group 3 for a federated model bears the least risk in all those respects because of the limited distribution and the controlled access to the database. Whereas it has been argued many times that security through obscurity is not real security, a point can be made that ignorance of the opponents decreases risks. This is another reason why the federated model may be more attractive for corporations and the government, and also from an ethical standpoint.

The proposal to involve lawyers was unanimous. It would seem necessary to have licenses, contracts and disclaimers for all parties involved in the data sharing. However, in an open model there are third parties that have not signed any agreements. Therefore they could: a) sue (libel, etc...) b) use the information for nefarious ends. The first issue can be addressed by legislation, which requires sustained, coordinated and powerful lobbying, difficult to achieve for an open model without the sponsorship of an organization. The latter can never be fully addressed from an ethical point of view, but like income tax, can be made part of living in the modern world as the global benefits are perceived to outweigh the disadvantages. This again requires legislation. As the more open model may get mired in legal difficulties, the parallel pursuit of two models is the best safeguard that some data sharing will actually happen.

Several participants indicated that they would be more likely to contribute if an institution like CERIAS, NIST or MITRE was willing to sponsor the efforts (i.e., expose itself first).

Conclusions

The workshop was successful in increasing the awareness and understanding of the issues that need to be resolved. The naming of vulnerabilities was resolved in practice by the MITRE CVE. However, the vocabulary still needs to be coalesced into a consistent set. The proposal of a common database schema was universally deferred, as it is a very demanding task. It was seen in part as the responsibility of the implementators of sharing models, and for the other part as something that would evolve from the attempts to share data. The maintenance of the databases, and how to insure the integrity and quality of the data, was discussed in the reports. Whereas delayed disclosure of vulnerabilities was suggested several times to limit the threat of vulnerability databases, it was unanimously concluded that legal strategies were necessary. As for the organization of common schemes, no single proposition satisfies all parties involved. However, we observe that the parallel pursuit of two has the best chance to achieve success.

Acknowledgments

Thanks to Tom Daniels, Mahesh Tripunitara and Russ Cooper for respectively writing the reports for groups 2, 3 and 4, in appendices B, C and D. Thanks to Jim Williams, Dale Blommendahl, Andre Frech, Steven Christey and Marc Dacier for sharing their notes on the deliberations of group 5, from which parts have been copied and edited. A portion of this work was supported by sponsors of the Center for Education and Research in Information Assurance and Security. Thanks to Kenneth Olthoff for comments on this manuscript.

References

1. Mann, D.E. and Christey, S.M. (1999) Towards a Common Enumeration of Vulnerabilities. Proc. 2nd Work. Res. Vuln. Data. 1999, CERIAS

2. Howard, J. D. (1997) An Analysis of Security Incidents On The Internet: 1989 -1995. Ph.D. thesis, Carnegie Mellon University

3. Krsul, Ivan (1998) Software Vulnerability Analysis. Ph.D. thesis, Purdue University

4. Krsul Andrade, I. (1999) Experiences in the development of the COAST vulnerability database. Proc. 2nd Work. Res. Vuln. Data. 1999, CERIAS

5. Olthoff, K. (1999) Thoughts on potential sources of error and bias in vulnerability databases. Proc. 2nd Work. Res. Vuln. Data. 1999, CERIAS

6. “MITS" Report: ISO/IEC TR 13335-1:1996. Information technology -- Guidelines for the management of IT Security -- Part 1. http://www.nssn.org/ and http://165.254.114.11/AnsiDocStore/

Appendices: Original documents from the 2nd workshop on vulnerability databases

Appendix A: The Open Model

Report for Group #1: The "Fully Available" Model

Facilitator:
Pascal Meunier - CERIAS, Purdue University

Participants:
Michael Hines - CERIAS, Purdue University
Frederic Dumont - CERIAS, Purdue University
Anthony Bettini - Netect Inc.
Janice Siersma - Dept. of Defense
Kevin Du - CERIAS, Purdue University

I. Mission statement

The mission statement given us was:

> "Assume that this database is completely open. Anyone can add to it, and anyone can access it. Copies of the database may be made and used in whatever manner desired. Analogy: GNU tool code."

In the first period, we were supposed to discuss and record the advantages and disadvantages to the model. Issues to be considered included the following:

- Ease of access
- Ease of update
- Consistency and integrity of VDB contents
- Intellectual property rights
- Commercial use
- Access control
- Who will keep it up to date?
- Who will want to use it?
- Who will want to hack into it?
- Who will want to shut it down?
- Fault-tolerance
- Data-mining suitability
- Entering historical information
- Expandability and flexibility
- Cost of maintenance
- Location
- Staffing
- Liability
- Use in investigation and in research
- Trans-national use, scale
- Timeliness of updates, redundancy, and longevity.

In the second period, we were supposed to consider what needs to be done first, second, and later to make the model work, who should define the architecture and schema, what should we start with, who is in charge and who funds it. Our work was supposed to culminate with a road map for how to achieve a sharable VDB according to the fully available model.

During the discussions, implementation solutions were proposed to maximize the advantages of the model, and minimize the disadvantages. In the last part of this report, I tried to synthesize the result of all the proposals into an implementation model that would receive the widest support and involvement, be morally and legally defensible, and provide the most usefulness for improving practical security and information security as a science while doing comparatively little in favor of criminals.

II. Advantages and disadvantages of the fully available model.

The fully available model provides the greatest use of access. Since by definition copies of the database can be made in whatever manner desired, there is no point in logging activity or authenticating users. This has the advantage that users of the database do not have to fear that the queries they make be a source of information to listeners; users may simply download the entire database and query it on a private computer.

The ease of update was in conflict with the protection of the consistency and integrity of the database, and all are implementation-dependent. An advantage of the fully available model is that by accepting "good" submissions from anyone, the chances are lower that the database entries relating to one problem present a one-sided view or are biased in favor of a particular group. The disadvantage is that each submission must be qualified as "good", otherwise the database could be used for 'slamming' , i.e., posting incorrect information to harm someone's reputation, etc...

The fully open submission mechanism followed by a trust-based filtering mechanism on the user's side, as proposed in Meunier's workshop submission, encounters significant practical difficulties, and is cumbersome. Yet we all felt that a control had to be implemented in order to improve integrity, reduce redundancy, and generally improve quality. We identified problems with allowing an author to modify or delete at will a database record. We agreed that the list moderation principle could be applied to the validation of initial submissions and every modification

of database records. It was also mentioned that the assignment by the author(s) of a trust level in the accuracy of the information they provide in their entry (as in Krsul's database) was a good idea.

The ease of access, entry and update all benefit from a standard format. A standard format for submissions, and another for updates may minimize the work done by moderators if they match or are a subset of a standard format for the records (schema). If the format of records follows a good standard format, then ease of use (as a sub-category of access) is increased. We took this idea further in the second period.

The intellectual property rights of the authors and of the database entity are different. The rights of the entity depend on the added value from the moderation. From the definition of the model, the authors have to either give the copyright to the database entity, or a perpetual license to all instances of the database. It was suggested that a licensing model (discussed later) could be used to incite users of the database to put back improvements into the database. It is both an advantage and a disadvantage of the fully available model that authors can't effectively withdraw their submissions once published, because the database may have been copied with these records.

The advantages of this model for commercial use is that it is free, it may contain information otherwise not available to commercial entities (from hackers or other sources), and provides a common ground of reference to compare commercial products and discuss vulnerabilities. The disadvantage is that for-profit entities may be reluctant to contribute, but they may be motivated to do so if it costs them less than to maintain their own database (and the database is good enough), the terms of user licenses may require contributions back into the database, and if preferential treatment is given to contributors (see later). In addition, the use of this database to establish a common format for the exchange of database information should motivate vendors to participate in the definition of the format, in order to enhance the suitability of the database for their later mining.

The access control for reading this database is inexistent. The disadvantage is that some people would like to have restrictions on the contents of the database for this very reason (e.g., we were plainly told that attack scripts in a fully available database are not considered favorably by the US government). On the other hand, this provides a lower overhead for the database in addition to improving the ease of access. An access control for adding or updating the database is possible without

violating the mission statement that "anyone can add to it", because the rejection of entries is based on a low quality of contents instead of authorship.

Maintenance of the database needs editors. An added cost is the training needs for those doing classification of vulnerabilities and attacks; however in the fully available model this cost is not borne directly by the maintainer(s), apart from the editors having to explain the rejection of some submissions. The advantage of the fully available model is that if someone lapses in the maintenance or does poorly, someone else may take over the database. The action of duplicating a database and starting a new maintenance operation is referred to as "forking" the database in the rest of this document.

We believe that everybody interested in vulnerabilities, including software vendors, consumers, security workers, news media, hackers, government agents, foreign governments, terrorists, etc..., will want to use such a database, whether or not everyone would be willing to admit it publicly.

Many people would want to hack into it, with the objective to disable it, to defame someone, remove embarrassing records, or to claim the "glory" of having done it. However, the non-confidentiality of this model is an advantage because it diminishes the possible benefits of hacking it.

We believe that anyone objecting to the public distribution of its contents would want to shut it down. This includes software vendors embarrassed by flaws and vulnerabilities entered in the database. The government might wish to shut it down (e.g., by removing funding) if the database contained attack scripts.

This database model would provide great fault-tolerance due to the ease of making non-confidential mirrors and duplicate copies. At the same time that additional mirrors provide more targets for attack, it makes it more difficult to disable or compromise the integrity of all targets.

The falsification and erasure of records in this database is very hard, because valid copies may be saved by anyone. Therefore it is the ideal medium for preserving historical information for later study. Someone pointed out that entering previous historical information would be a good way to get this database started.

The expandability and flexibility of the fully available model is somewhat function of its implementation. However, because anyone is allowed to make a copy and transform it, this model has the potential to be the most expandable and flexible.

The advantage of this model is that the cost of maintenance would probably be the lowest, but a disadvantage is that the funding of this maintenance has to be sought. Someone proposed to sell banner space for web browser access to the database, and that giving large space to acknowledge the sources of donations might help.

This database model is mobile and forkable, and therefore can probably use any location. The staffing is function of the location.

The liabilities of a fully available model are potentially the greatest. However, suing the maintainers of the fully available model may be difficult because the database may be attached to a "front" corporation that could fold while someone else takes over the database. It was suggested that various disclaimers be attached to the use of the database, and that the interface to the database would enforce an acknowledgment of the terms of use, and the use of insurance. In the end, it was obvious that we needed access to a lawyer to plug the legal vulnerabilities of the database.

The fully available model is ideal for investigations and research because it provides a common set of data to discuss and on which to reach agreement. It also provides an ideal medium for the exchange of discoveries, and their validation or improvement by the examination of peers. Because it can be copied, intensive work on the database may be done elsewhere than on the primary server, and therefore the service to other users needs not be degraded. It is also easy to archive (an important aspect of comparisons and studies). We also talked about the need for a good search interface.

The trans-national use of this database may be viewed as a problem from the point of view of national security. However, the exclusion of attack scripts, when possible, would seem an acceptable compromise.

The timeliness of updates is basically a function of who is maintaining it. The advantage of this database is that if someone is dissatisfied with the way this is done, then it is possible for a better copy to be made.

The scalability of the database is implementation dependent. This model may become highly redundant if different competing instances exist. The longevity of this database may be the longest. It is possible for such a database to become dormant for some period of time, until someone else picks it up again, so there is always the possibility that it's not really dead even when it would appear so.

An additional issue we identified with this model is "boot-strapping", i.e., reaching enough momentum, quality and interest that the database becomes viable. We talked of soliciting Bugtraq, NTBugtraq for entries with the goal of populating the database, and obtaining enough funding to quickly increase the initial contents of the database.

III. Implementation model

1. Basic organization

We agreed that the list moderation principle could be applied to the validation of database records. This presented the advantage that human judgement could be applied to control some other issues. The principle mechanism is shown in Fig.1. It is in essence identical to a board of editors that reviews submissions in their own area of expertise. Submissions must adhere to standard published formats and be complete to the point where the workload of the editors is limited to accepting, rejecting, delaying or requesting improvements to a submission before it joins the database. The submissions should be in a format that makes their addition to the database straightforward.

The database is envisioned as a "distribution" assembled in a central location for the purpose of its maintenance and administration, with mirrors elsewhere for robustness, distributable on cheap optical media (e.g., on CD-ROMs or DVD-ROMs). However, it is also viewed as fully downloadable, and always available such that anyone can set a copy on a different server. It is hoped that by this mechanism, any group of individuals who feels can do a better job of administering the database and setting up policies may do so at will, thereby ensuring that the best version of "children" open databases survives and adapts. Therefore, the responsibility for the viability of the mature database lies with the editors, acting as the trustees of the database, including the solicitation of submissions.

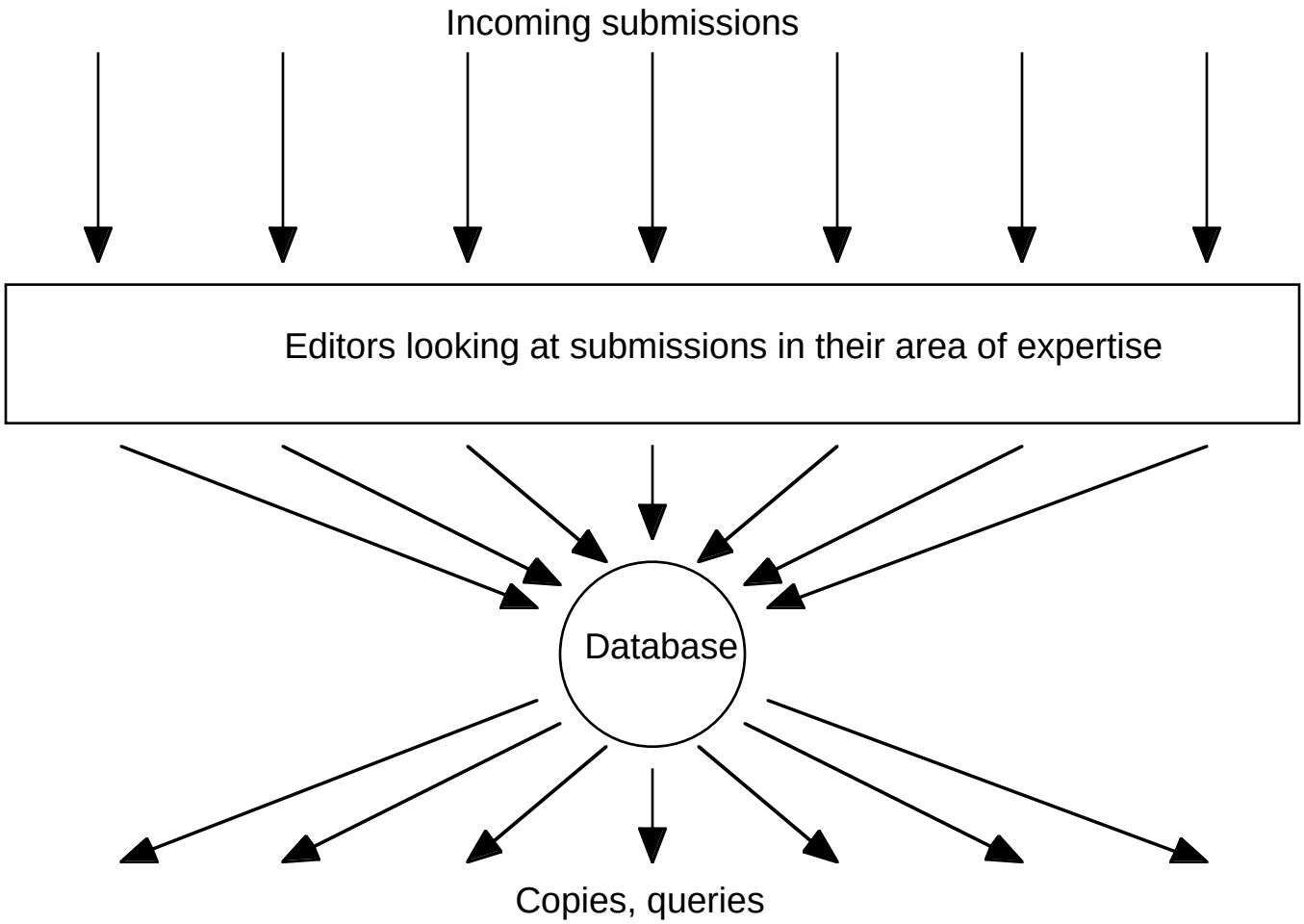

Figure 1. The open database's proposed mechanism of operation.

2. Schema.

The schema of the open database is a deciding factor for its usefulness, and therefore the support it will get. The desire was for this database to be useful for the widest number of legitimate users. It was impractical for us to decide upon a fixed schema that would answer the needs of everyone. Therefore, we decided upon an open, extensible schema model. The attractiveness of this model is that every existing party as an interest in participating, in order to improve the future usefulness of the database for themselves. The extensibility of the database would also mean that additional fields could be easily added for private use by anybody mining the database. XML was proposed as a tool to define database records and submission formats. The set of record and submission formats would be referred to as VML, for Vulnerability Markup Language. It was our expressed hope that the construction of the database would provide an evolving standard format for the sharing of vulnerability data, to be submitted to the IETF.

It appears most useful that the VML should include fields that would allow security problems to be viewed in multi-dimensional aspects. Dimensions

deemed significant in viewing a vulnerability database were by attack, by resources affected, by core vulnerability, by policies affected, as well as by environmental properties. These do not prevent or limit the use of keyword searches or fields like general descriptions or comments and analysis. The language defined by group 5 would form the basis of the inspiration for naming tags. It was suggested that some data from the CERIAS database be used to evolve a group consensus on the format, and that the data be used to provide an initial population to the new database. Another suggestion was that the vulnerabilities discussed by group 5 should be used to shape this new format, because group 5 will already have discussed the correct vocabulary to describe the vulnerabilities. It was proposed that a mailing list be setup for the purpose of shaping VML during this initial translation, in order to respond to the interests of most possible contributors.

IV. Policies

The following are a set of nucleus guidelines for the moderators of the database. These guidelines were decided upon as a consequence of the mandate, but were viewed as flexible in response to feedback from all concerned parties.

1. Courtesy notices. In the event that a vulnerability which could severely impact existing assets is discovered in vendor software, it appeared wise to first notify the software vendors so they could try to prepare a response or at least have a lead in solving the problem. The suggested time frame was 48-72 hours, and subject to the exception that if a work-around is already known or a need for warning is present (e.g., attacks are in progress), the presence of the vulnerability would be released immediately. In any event the author of the submission would be notified of the action taken.

2. Scripts. There is no requirement for scripts (exploits) in a submission. If a script is submitted, it should be a detection script instead of an attack script. If there is no alternative, then an attack script would be released only if a work-around or patch is available. An exception to this would be granted if the vulnerability has been present in the database for longer than a reasonable time-out period. It is suggested that the time-out period be 2-4 weeks, subject to the editor's judgment.

3. Recognition for public duty

It is requested that all users and indirect beneficiaries of the database recognize the work of the editors and authors, and give them appropriate credit during performance reviews. We aim that for editors, the work be

recognized on an equivalent level to being an editor of a conventional publication in their field of expertise. Nevertheless, we realize that this recognition will mainly be earned from the reputation and usefulness of the database, which will further motivate the editors.

4. Initial submission formats

Full Vulnerability Entry, Addendum.

5. Licences

It was suggested that in order to generate up-to-date submissions, increase the quality of the database, and the sense that contributing to the database will yield future benefits for the authors, the database should adopt a modified version of the GNU public license to apply to comments and analysis but not code (e.g., open content, <http://www.opencontent.org/>). The purpose of this license would be an obligation (at least moral) to put back in the open database or open domain any improvement made to a record. The choice between this license and a "free use" license would be decided by the authors of the vulnerability entries.

The licenses for users aims simply at protecting the authors, editors and trustees of the database and at soliciting future improvements. It was suggested that the user license contain terminology such as "use at your own risk", "I agree not to use this information for criminal purposes" and a "click here if you agree" interface.

The license for forkers aims at preserving the open nature of the database. The license for authors aims at preventing withdrawals of database entries, in effect granting the open database and all its children a perpetual license for the use of the submitted information, while preserving the other rights of the authors and limiting the liabilities of all involved parties.

6. The editors should make sure that a snapshot of the database is archived every so often (no period was specified during debate).

V. Plan of Action

1. a) Find a leader/starter. It was suggested that the initial gestation of the database be done in an academic environment because it is more neutral, has fewer chances of encountering conflicts of interest and won't compromise corporate public images and policies or be affected by them. This opinion is does not seem to have any fundamentally robust basis, but appears more practical.

b) With the leader there should be a lawyer to review the consequences of the policies and establish a solid legal foundation for the open database.

There is a need for helpers for the initial translation and population of the database, as well as initial material and funds. Starting money for the database is partly available from CERIAS and its sponsors.

2. Decide schema and standard submission formats. Setup the mailing list for this purpose and VML v.1.0, hopefully with company involvement because it will mean less work for them in the future for using this database, and possibly a standard information exchange format.

3. Get money, major equipment and materials. Formulate policies for editors. Write the licences for users, authors, and "forkers". It was suggested that the database interface acknowledge donations (equipment and money) and may raise additional revenues by selling publicity on the basis of the number of hits.

4. Add moderators and mature the database.

VI. Conclusion

We view the schema of this database as an important contribution irrespective of the value of the data in the actual database, inasmuch as it helps define a common format for sharing vulnerability information. In this regard, the process of generating an open database is as important as its contents and use. We tried to provide as many incentives for participation as possible. However, we also realized that such a database can be a useful resource even if not everyone participates fully. Moreover, there are already people who make their discoveries public, and who might not participate in a more closed model; this database presents a fitting mechanism for the release of their findings.

VII. Addendum

After the report on the work of this group, it was pointed out that there were significant resemblances with our proposal and an abstract presented at the UC Davis Intrusion Detection and Response Data Sharing Workshop: "Designing IDLE: The Intrusion Data Library Enterprise" by Lindqvist U., Moran D., Porras P.A., and Tyson M. The resemblances validate the common choices.

Appendix B: The Centralized Model

Report for Group #2: Centralized Model

Facilitator:
Tom Daniels - CERIAS, Purdue University

Members:
James Davis - Iowa State University
Matt Cerha - Trident Data Systems
Amy Hammond - Dept of Defense
Todd Johnson - Bureau of Public Debt
Yasuo Miyakawa - Info Tech Promotion Agency
Aaron Schwartzbard - Reliable Software Technologies
Mowgli Assor - Ohio State University

1. Architectural Organization

The group's first goal was to clarify the notion of a centralized VDB (CVDB). The group decided that a CVDB would entail a database of vulnerabilities updated/modified in one location but accessible by way of the Internet and perhaps other networks. We decided to not focus on engineering details unless there was some question of the project's technical feasibility. The CVDB would be managed by some central organization that would be in charge of defining the schema, data review and consistency, funding, and policy. To ensure that the needs of the community were being met, it was decided that an oversight process for the organization must be formed.

Centralized databases are well understood technically and managerially. A CVDB would naturally benefit from this. Compared to a distributed database, consistency of the database would be easy to enforce. Some misgivings were raised that the CVDB would create a bottleneck, but it was agreed that a VDB would probably never exceed the size of very successful centralized databases currently being served to the general public such as Amazon or Yahoo.

Management of the CVDB would be handled either by an organization specially created for the job or an existing organization willing to accept the task. For convenience, we will call this organization the managing organization or MO. We seemed to agree that neither government agencies nor vendors were suitable because of conflict of interest and trust issues. This leaves either a special non-profit organization or an academic organization.

To ensure that the MO maintains the database in a way that meets the needs of academic, commercial, and government use, we decided that an oversight committee should be formed. The committee would set policies for the MO and provide oversight.

2. Policies

Maintenance and creation of any database requires well defined policies and procedures. We discussed several different policies that fundamentally shape what type of database would best serve everyone. These policies cover data disclosure, data submission, and data review. Other topics were data content and policies concerning methods of user subscription and authentication.

The data disclosure policy was one of the most discussed aspects of the CVDB. Most members agreed with eventual full disclosure of vulnerability information to the public. Some members felt that vendors would be more likely support a VDB that gave them some measure of prior notice. One possible scheme was that attack and other highly detailed information be withheld for 30 days to allow vendors time to correct or otherwise act on the problem.

Everyone agreed that anyone should be able to submit information for inclusion into the database. This includes the hacker community, vendors, etc. Of course, it would not be wise to allow everyone direct data entry so we agreed that an *a priori* data review scheme would be needed. In this way, changes and new submissions would be put into a queue that would be inspected by reviewers of the MO.

Data content was also discussed during the session. The fundamental issue seemed to be that of releasing attack information. Most members had no problem with the release of freely available attack information assuming there were no legal problems with copyright issues. Restricting access to attack scripts, etc. was discussed, but the question then becomes what information is required to exploit a vulnerability. It may be that the information sufficient to uniquely identify the vulnerability makes its exploitation trivial. Thus we had no specific resolutions for this issue

3. Advantages

The advantages that the CVDB has over a distributed model were fairly

clear:

- Review and data quality control are easy.

- Access and Update of data are simplified.

- A CVDB can be more easily kept stable and consistent.

- Policy matters are simplified because a single organization is in control.

4. Disadvantages and How We Addressed Them

Scalability may be an issue, but we did not deem it a critical one. Even if the CVDB had 10,000 records (considered unlikely), many centralized online databases are much larger and of much greater interest to the general public and despite this, they suffer few scalability problems.

Organizational bias. This is the reason for the oversight board proposed above.

Availability. A CVDB is a target for attacks and prone to a single point of failure. However, access to the information can be maintained through data mirroring and by using a "hot spare" contingency plan.

5. Things that we spent time on

Open vs. Access Control

Copyright issues -- we need a lawyer.

Vendor Participation -- It is important to try to build something respected by vendors while maintaining an unbiased or neutral position. Delayed release seemed to be a reasonable solution, but only if the delay period is fixed.

Appendix C: The Federated Model

Report for Group #3: the Federated Model

Facilitator:
Mahesh Tripunitara - CERIAS, Purdue University

Members:
Balinsky, Andy - Cisco Systems
Grance, Timothy - NIST
Marez, Greg - Army Research Lab
Markham, Thomas - Secure Computing Corporation
Morrison, L. Kent - Army Research Lab
Nelson, Angi - Cisco Systems
Richardson, Tom - Iowa State University, Rockwell Corporation
Soliz, Phil - Trident Data Systems

1. Introduction

This report summarizes the discussions held by the working group on the Federated Model for vulnerability databases. This document is written in the style of a paper or an article. It does not attach statements to any particular person from the group. Most, if not all, assertions or decisions were by consensus and therefore may be attributed to the entire group. Where there was disagreement, that fact is noted in this report.

1.1 Organization

This report is organized as follows. In section 2, we provide some definitions and discuss the starting assumptions. In section 3, we deal with the model and an architecture to realize it, and advantages and disadvantages with respect to several issues. The issues are primarily drawn from the hand-out from Prof. Spafford that was intended to aid the discussion. We summarize and conclude in section 4.

2. Definitions and Assumptions

A federation is a loose union of several distributed entities on an issue. In the context of this report, "distributed entities" refers to governments at any level, governmental institutions, commercial establishments, colleges and universities, and perhaps individuals (note: the group expressed ambivalence over what an "entity" is.)

The "issue" is security vulnerability databases. We assume that all entities involved have a common definition for "security vulnerability" and "security vulnerability data." The latter populates the database. The "loose union" is a common schema and the semantics associated with that schema.

"Schema" is used in the sense of a relational database: it defines the structure with which data is stored in the database. A schema involves labeling of fields. The semantics associated with the labeling (and therefore the schema) are also part of the loose union.

Each entity involved in the federation "owns" a database in which data is stored structured by the common schema. Each database may also contain proprietary extensions to the common schema. The proprietary extensions are not visible outside of the entity that owns the database. The schema also contains extensions for "general members." This is discussed in section 3.1, which also discusses the notion of general membership.

Thus, this report does not discuss details on the schema or the semantics associated with the schema. This is left as future work under the federated model. This report focuses on the federated model and its architecture.

3. The Federated Model, and its Advantages and Disadvantages

This section discusses the model and the advantages and disadvantages with it. The discussions are based on several perspectives, including ease of access, maintainability and ease of update.

The federation that maintains the vulnerability data has two parts: a distributed part and a centralized part. The distributed part comprises of the databases and their owners. The centralized part comprises of a body that sets standards for the federation and provides a common interface for access to the data in the databases that are part of the federation.

3.1 Entities that Access the Databases

We have two classifications for entities that access the databases. Each classification groups the entities that require access into two groups. One classification is based on ownership of each database. This classification typically applies in the case of writes ("write" is synonymous with

"update.") In this classification, the entities that require access are classified into owners of the (each) database, and others.

The second classification is used typically when discussing read access. This classification groups the entities that require access into general members and the public. The next section that discusses the organization of the federation defines "general member."

The context of the discussion should clarify which of the classifications is being adopted for that discussion.

3.2 The Federation

The federation has three types of members: The Executive Panel, the General Members and the Public. We discuss each of the three groups in the context of the costs in maintaining their status in the federation, and the benefits and responsibilities that arise from holding such a position.

The Executive Panel is a subset of the general members and is elected by vote by the general members. The cost for being a member of the executive panel, in addition to the costs for being a general member, is that the executive panel maintains the central interface to access the vulnerability data in the databases, and the executive panel must institute committees to look into various technical and other matters that relate to the federation, from time to time. For instance, a periodic review of the schema would be the task of one such committee.

The benefits with being in the executive panel are that the executive panel has administrative oversight over the direction and status of the federation. Thus, the executive panel potentially has more input into the agenda for the federation than the rest of the general membership.

A general member pays money to be part of the federation. The amount is based on the annual revenues and other such financial aspects of the general member. For instance, a small corporation may only pay $2000 to be a general member, while a large one would pay $10000. Every general member must represent a sizeable constituency.

The benefits with being a general member are that the general member has access to certain extra information from all the databases. For instance, exploit tools to test for the existence of a vulnerability. The general member also gets easy, perhaps free, access to discussion forums and has voting rights in the federation. A general member may also get earlier

access to information on vulnerabilities related to their products than others. For instance, the general member in whose system a vulnerability is reported gets 48 hours before the existence of the vulnerability is publicized.

The public has free read access to the databases. The public does not have access to some of the fields from the database that only general members have access to.

3.3 Ease of Access and Updates

The federated model is second only to the balkanized model in terms of ease of (read) access. The owner of a database has easy read access to their database. The central interface allows for access to the other databases in the federation.

Multilevel access control is employed for read access. This could be role-based. Section 3.2 that discusses the federation, mentions the difference in access privileges of general members and the public.

Updates are more controlled than access and a validation process is used to ensure that the data in the database is "good." The validation could be done in a centralized manner, which means that the executive panel would appoint a committee to scrutinize each suggestion for an update.

Alternately, the validation can be the responsibility of the user that reads the data from the database. In this case, a "validation" field is part of the schema and the user can use the value of this field to decide how much faith he or she wants to place on the data. The value in the validation field is decided by the entity that performs the update.

Anyone may suggest updates (note: there was considerable disagreement on whether to adopt a centralized update policy, or a distributed update policy.)

3.4 Consistency and Integrity of the Contents

If the update is centralized, the consistency and integrity issues are dealt with at the time of update. That is, any database that contains the data has the same entry as any other database.

If the update policy is distributed, each database may have alternate entries for similar or the same vulnerability. When a user requests a read

access from the centralized interface, a mechanism is in place to fetch the data for display from the databases that may contain data relevant to the query entered by the user. If there are similar entries in several of the databases, a "merge" is performed on the data and the composite is presented to the user (note: this issue was left for later discussion, but was not revisited owing to lack of time.)

3.5 Intellectual Property Rights

An individual or organization that owns the copyrights to a piece of data that is entered into the database relinquishes some of their rights, but not all, to the federation. Similarly, a user performing read access on the data is granted some rights with respect to use of that data, but the federation still maintains its rights on the data (note: the consensus on this issue was that the lawyers would work it out!)

3.6 Updates, Maintenance, Location, Staffing and Funding

Some of the funding will come from the membership dues paid by the general members of the federation. The other issues are unresolved.

One of the options is for some of these tasks to be performed centrally. The characteristics necessary in this central part are as follows: It must be neutral. It must be competent. It must be able to engage in contracts for the federation and accept money on behalf of the federation.

3.7 Trans-National Use

Trans-national use of the data and the databases is allowed. The membership criteria of the federation subsume any trans-national issues in terms of membership or access. The federation will do its best to ensure that no national or international laws are broken in the formulation of its policies and rules, though it will be unable to make such guarantees.

4. Summary

The federated model has the best chances of success from amongst all the options. It retains both central and distributed characteristics that are necessary in the creation and coordination of such databases. The common schema is a crucial and attractive feature. Commercial establishments will find this model attractive as there are opportunities for profit making. The open model is unattractive to vendors, while the centralized model breeds mistrust. The balkanized model, which is the status quo, is

unacceptable and ineffective.

Commercial entities such as firewall vendors could organize unto themselves. That is, constituencies with similar concerns and issues might easily form under this model.

Several important issues were not discussed at the meeting. These include:

- Who will want to use the databases;
- Who will want to hack into it or shut it down;
- Fault tolerance;
- Data mining;
- Entering historical information;
- Expandability of the databases and the model;
- Liability issues;
- Issues of scale and longevity.

The following roadmap or "to do" list was adopted to continue the efforts of the federated model working group:

1. Quarterly meetings
2. Draft charter and bylaws
3. Design the federation
4. Design the schema
5. Design the sharing mechanisms
6. Perform initial populating of the databases

Appendix D: The Balkan/Status Quo Model

Report for Group #4: "The Balkan/Status Quo Model"

Facilitator:
Russ Cooper

Contributors:
Adam Shostack - Netect Corporation
Tom Truden - Ford Motor Company
Stephanie Miller - CERIAS, Purdue University

Other members:
Dave Bailey - Arca Systems
Eli Chandler - Dept. of Defense
James Donndelinger - The Aerospace Corporation
Michael Gilmore - US General Accounting
David Mann - Mitre Corporations
Kenneth Olthoff - Dept of Defense
Steve Romig - Ohio State University
Thomas Truden - Ford Motor Company

1. Mission Statement/Purpose

The group began with the intent of reasoning why Balkan database models exist and what can be done to move away from them toward a more publicly profitable model. As Adam Shostack presented to the general forum, the discussion essentially can be divided into three sessions: Why Balkanization Occurs, Mitigators to encourage movement away from the Balkan model, and Next Steps. This report has been formatted such that it can be easily "plugged in" to one of the other proposed models.

2. Balkanization Defined

A. Causes

I. Fear of Bad Publicity

There is a general fear among business and vendors that bad publicity would result from what the public may view as contributing to the "hacker" community.

II. Classified Data

Some data which goes to certain areas of the government never returns.

III. Fear of Liability

Companies do not want to take the chance of being sued because a hacker caused damage or bad publicity to another entity, and the company's name is associated with the site where the hacker got his tool kit.

IV. Copyright issues

V. Increased Accuracy of Private Databases

There is a perception that private databases, under the control of the organization, contain information that has been more rigorously vetted.

VI.Investment

Some companies have already spent a lot of money building private databases.

VII.Control of Data

There are a lot of availability, control, and bureaucratic concerns that are easier to handle in-house than through coordination.

VIII.Ability to Tailor

Local databases are designed and modified to handle local requirements. Local databases can also evolve more quickly than shared ones, as schema can change for organizational need.

IX.Unwanted disclosure

Companies do not want information leaking before an advisory or patch is released, plus can gain interesting information from query analysis.

X.No Established Sharing Mechanism

No easy, trusted mechanism exists which enable companies to contribute.

XI.Perception of Advantage

The perception of 'my database is my advantage'.

XII.Production Control

Use of a database to control production processes requires a database under local control, and contains much business process information which is not going to be shared.

B. Mitigators

I. Fear of Bad Publicity

a. A Respected Sponsor such as NIST, CERIAS, or NCSC can allow an organization to point to standard practices.
b. A common practice, even without a sponsor, can gain recognition. (GNU model).
c. Industry Organizations such as CARO could provide cover.

II. Classified Data

This is unlikely to change, but a Freedom of Information Act Request might mitigate it somewhat.

III. Fear of Liability

a. Database content review - Review by a competent authority.
b. End User Licenses - Authority to modify the database.
c. Multiple Use Rules.
d. Restricted Access Rules.
e. Legal Defense Fund.
f. Indemnification for submitter or repository.
g. Anonymous or pseudonymous posting.
h. Clarification of Duty to vendors/creators of OS - This will head off any problems with legitimate software companies.
i. New laws.

IV. Copyright issues

a. Care must be taken to ensure that permission is gained from legitimate software companies, and that fair use practices are observed.
b. A GNU-ish license for work contributed.
c. Funds would be required to purchase licenses and gain permissions.
d. A model of pointing to commonly available data on the web instead of copying the data would avoid the copyright issue.

V. Increased Accuracy of Private Databases

This could be mitigated by moderation, peer review, a clear system of quality standards, a right of rebuttal (credit databases are the example here), verification (funds would be required), and research contributions by respected institutions.

VI. Investment

Sunk costs could be balanced by:

a. A business case demonstrating that the shared costs of a common database would lower internal costs.

b.Recognition for financial sponsorship.

VII. Control of Data

a.Mirrors, backups, compact disks of databases would alleviate concerns about availability.

b.A common database would provide increased perceived value of having a database to which many experts contribute.

c.A common database would provide companies an opportunity to expand their sphere of control and range of infighting.

VIII. Ability to Tailor

a.Make database simple.

b.Use a common enumeration or terminology.

c.Flexible import/export mechanisms make it easier to get data into individual format.

IX. Unwanted disclosure

a.Partial Disclosure (control, limit contributions).

b.Distribute copies of database to allow queries to be local.

X. No Established Sharing Mechanism

This will need to be resolved based upon which common model is chosen.

XI. Perception of Advantage

a.High flexibility of common database.

b.Quick response time of common database.

c.No requirement for complete sharing.

d.Change area of competition (eg, virus market, other "mature markets").

XII. Production Control

There are really no mitigators here, as a database used for this purpose will contain much non-shared proprietary data.

C. Next Steps/Recommendations

Our recommendations include the following:

A. Incorporate the mitigators from this report into the execution plan for whichever database model is chosen in order to encourage organizations to begin contributing to a common model, moving gradually away from

exclusively the Balkan model.

B. Adopt either a full schema (such as the Central, Federated, or Open models), or

C. Adopt a CVE model (as proposed by MITRE) with the starting number of CVE-10001 for the first correlated database record. CVE-1 through CVE-10000 have been reserved for all bugs/exploits starting with the "Moth in vacuum tube" until the present.

3. Conclusions

From the participants present it became obvious that the Balkanized model will not go away. Despite our original mission statement to see whether it would be possible to eliminate it, we realized that there are just too many real benefits to having your own VdB.

Further, it was commonly agreed that no single repository would be able to serve the myriad of purposes the Balkans have for their own repository. Therefore, we agreed it made sense to promote two models working in harmony.

The Balkans were initially seen as the holders of great information whom, for various reasons, would not or could not share such information with others. This turned out not to be the case. When offered the opportunity to contribute "some or all" of their information, we found that everyone present was prepared to participate. The mitigating factors listed should not be under-estimated in their power to enable Balkans to contribute to a larger effort.

With a modified Balkan mentality that allows for sharing of some information and broad participation and publicity in a shared repository, its felt that the goals of the workshop could viably be achieved.

The Balkan group looks forward to the proposals from the other groups to see which model works best for sharing whatever information is shared. Expect the members of the Balkan group to voice their opinions enthusiastically towards the effort.

Appendix E: Fundamental Questions

Report for Group #5

Facilitator:

Marc Dacier	- IBM Research

Members:

Bob Abbot	- Abbott Computer Partners
Dale Blommendahl	- XYPRO Technology Corporation
David Bonewell	- NCR corp.
Patricia Burt	- Dept of Commerce
Jim Carrales	- Kelly Air Force Base
Andre Frech	- Internet Security Systems
Al Holt	- Network Intrusion Analysis
Steve Kos	- L-3 Network Security
Eric Meijer	- Harris Corporation
Peter Mell	- NIST
Samuel Moore	- EWA-Canada LTD
Douglas Moran	- SRI International
Mike Prosser	- L-3 Network Security
Aaron Rhodes	- CISCO systems
Allen Wilson	- Prudential
Diego Zamboni	- CERIAS, Purdue University
Kevin Ziese	- CISCO systems, Inc.

A) Terminology by Jim Williams

Group 5 was given three key questions. We began with the third.

Question 3. Should Vulnerability DataBases (VDBs) include attacks, incidents, tools, patches, etc.?

Answer 3. Yes, all of these, plus system component identifications, system and organizational policies, asset valuations, and probabilities of exploit.

Rationale. Our subgroup concurred on the answer, and we were able to provide several justifications.

R 3a. All of this information is useful, and it should be included even if some items are of interest only to some users.

R 3b. Including related concepts in a VDB allows richer linkage and thus better utility. For example, one may want to query on attacks to find related vulnerabilities of interest.

R 3c. Due to apparent diversity of opinion on the content and structure of VDBs, it is important for them to be empirically grounded. That is, they should be grounded directly or indirectly on raw incident data, including any information that may be needed to assess seriousness (e.g., tangible loss, frequency of exploit, available patches, identification tools).

Question 2. What fundamental issues need to be resolved?

Answer 2. We need a language. Specifically, we need to be able to make relevant distinctions clearly and conveniently.

Rationale. Our evidence for this is that almost every discussion involved unintended misunderstandings between well-informed participants. In most cases, these confusions were the result of unnoticed overloading of terms. We simply have more things to talk about than we have terms to describe them. Key examples follow.

R 2a. The word "vulnerability."

In a specific application, instance or use of a product: a

vulnerability is a feature that allows harmful, unintended behavior.

In an product prior to use: a [potential] vulnerability is a feature that may be a vulnerability in some intended application, instance, or use.

Notice that the first definition doesn't work for products because not all consumers of a product will have the same intention or the same understanding of harm. For example, a video screen that can be seen by anyone may be an asset in an airport as well as a liability [i.e. vulnerability] in a classified environment. A vendor who wishes to sell video screens to both communities may need to regard ease of viewing as a potential vulnerability.

R 2b. Policy

The above discussion of vulnerabilities manages to avoid the word "policy." In some cases, intended, harmless behavior may be obvious, in others it may be codified by policy documents. The Common Criteria (CC) allows for both situations.

In those cases which are policy based, there is the immediate question of what "policy" means. One possibility is that of an organizational security policy. The other is that of a technical security policy. The latter assumes more layers of prior analysis. Typically, a technical policy is invented in response to a corresponding organizational policy. These two kinds of policy are enough different as to cause misunderstanding whenever they are confused. In fact, this was one of the first confusions to be unraveled in the construction of the CC and its predecessors.

R 2c. Atomicity of incidents and attacks.

For some, an attack is an atomic event while, for others, it is a sequence, scenario, or pattern of events. The same is true of the word incident. Howard's thesis illustrates both perspectives by treating an incident as a sequence of attacks.

R 2d. Abstraction level of attacks.

We found it useful to consider at least three levels of abstraction:

The concrete level: an attack instance [or incident] at a particular time and place. Concrete attack instances are what get reported to CERT (although some sanitization may occur as part of the reporting process).

The "portable" level: a sanitized attack is the set of all repetitions of a concrete attack instance. Information that is not of general interest is filtered out, e.g., time and location of the attack, details about the extent of tangible loss, and perhaps details of the system configuration under attack as well. Although we did not come up with a precise definition of "all repetitions," we believe that this level is well understood in practice.

The reusable-information level: an abstract attack consists of the common security-relevant characteristics of a set of similar attacks. An abstract attack is normally platform independent. We believe that the community does not yet have a general ability to present attacks at this level because the community hasn't yet converged on the salient, reusable attributes of concrete attacks.

R 2e. Response to attack. The Common Criteria (CC) and the "MITS" Report (ISO/IEC TR 13335-1) use the terms "countermeasure" and "safeguard," respectively. Interestingly, these two words have different meanings in different communities.

To some military folk (but not in the CC), a countermeasure is considered to be a preventive measure, whereas a safeguard always includes the broader alternatives of attack identification and attack recovery.

To many civilians, the term countermeasure has a hostile connotation and is thought to be analogous to counterattack.

R 2f. The word "system" has at least two good uses in connection with attacks:

Sometimes the system is the target of discussion. The CC uses the term Target of Evaluation (TOE) in preference to system in this sense.

Sometimes the [whole] system is the big picture that contains the

target of discussion as well as its users and relevant aspects of the target's environment. The CC uses the word environment to refer to the whole big-picture system, something that takes a little getting used to for some people.

R 2g. People use various terms with multiple overlapping meanings for what an attack might compromise. The root words in these terms are resource, asset, and information.

For some people, assets are normally monetary in nature. Most people agree that information is a type of resource. However, the CC defines assets as information and resources. For sake of discussion, let us take resource to be the general term that includes information and monetary assets, among other things. What resources are relevant in discussions of attacks and vulnerabilities? This question tends to generate a surprising amount of heat. Without answering the question, lets at least look at some categories of resources:

IT resources (e.g., information, processing capabilities)
non-IT resources (e.g., money, property, life, liberty)
target resources (part of the target of discussion)
environmental resources (outside the target)

In conversation, people often equate resources with IT resources or with target resources. All but one of our group agreed that all kinds of resources are of potential interest, at least at the level of concrete attacks.

A telling example was a device that talked with satellites. One could issue a sequence of commands that caused the satellite to go off line, at which point the cheapest recovery strategy was to launch another satellite -- at a cost of about $22 million. In this concrete example, the monetary data provides a clear understanding of the attack's significance. In the context of a more abstract attack description, the main point might be simply that attacks which necessitate remote maintenance are good candidates for preventive safeguards.

Question 1. How can we develop mechanisms to promote standardization of terminology, classifications, schema, storage, etc.?

Answer 1. Regarding terminology, see our answers to Questions 2 and 3.

In addition, we strongly recommend "stealing" terminology from other sources when feasible. Other good sources include the CC and the MITS Report, as well as interested professional groups, e.g., security auditors, security consultants. Of course, we should also take a look at the results of the first workshop on VDBs.

Time did not permit detailed discussion of other fundamental issues in our subgroup, but we believe that they will be more easily resolved once basic terminology has been agreed upon.

B) Naming Schemes by Steven Christey

The Naming Schemes subgroup discussed various issues related to defining a naming scheme. There was some discussion related to the common naming convention and broad classification utilized in virus databases.

There was some debate regarding the merits of an arbitrary naming convention (e.g. by assigning simple numbers to vulnerabilities). However, most of the group advocated a principled naming scheme that would be consistent and repeatable, where the name itself gave some indication as to the nature of the vulnerability. The group then focused its efforts on defining the characteristics of such a naming scheme.

The following requirements for a naming scheme were identified:

1) It should avoid reliance on taxonomic information and focus on easily observable features, e.g. names (of the application, service, protocol, and/or vendor), etc.

2) It should nonetheless have the basic characteristics of good taxonomies, e.g. repeatability, specificity, exhaustiveness, and mutual exclusiveness.

3) When deriving the name, the exploit should be separated from the inherent vulnerability where possible.

4) Aliases for the standard name should be supported, e.g. to recognize "well-known" names for vulnerabilities such as Smurf or Ping of Death.

It was proposed that a mailing list could be created for discussion of the assignment of names to new vulnerabilities. Security tool vendors could then match their tools to the assigned names.

Sample Naming Convention
- -

The working group spent a significant portion of its time in developing a simple naming convention. The proposed convention divided the name into several components. The first component of the

name identified the broad category of the problem. A proposal was made to have the first component identify whether the vulnerability was in the Network or on a particular Host, but the group ultimately decided to base the first component on the "location" of the vulnerability (i.e. protocol, OS, application, policy, or services). Additional components of the name identified progressive refinements of the category. The group attempted to name three different vulnerabilities to test the proposed convention:

Session hijacking: Protocol->TCP->Authentication->Hijacking
SSH insertion: Protocol->TCP->Application->SSH
Smurf: Protocol->ICMP->Specification->Smurf

Problems with the sample convention

One problem involved the degree of specificity of the vulnerability name. It became clear that in some cases, very similar vulnerabilities might be assigned the same name. To avoid this duplication, an arbitrary, non-repeatable name might need to be assigned in the last component. A separate problem arose in consideration of the name for the SSH insertion vulnerability, in which SSH could be thought of as either a protocol, service, or an application. Additionally, the meaning of the terms used in the name had to be made explicit; for example, did the term "specification" as a modifier to "protocol" refer to an incomplete specification, or a flaw in the implementation? Finally, the group debated whether "missing" (non-applicable) components should be explicitly recorded in the name, so that the Nth part of any name always referred to the same component. For example, the Smurf vulnerability does not apply to operating systems, so it was suggested that a blank be placed into that component of the name,
e.g. Protocol->ICMP->Specification->blank->blank->Smurf.

It has been decided to continue i) the exercise with more vulnerabilities once the participants would have access to their respective source of information and ii) the discussion regarding the best way to give names to the various parts of a 'name'. A mailing list was created for this purpose.

More general problems with naming conventions

A practical problem arises if a naming convention is adopted by the community. If tools advertise their capabilities using the convention, then hackers would be able to easily determine which vulnerabilities are not identified by the tools, then design attacks that exploit undetectable vulnerabilities.

The question was posed as to what would be a good business case for adopting a naming convention. ID vendors mentionned that they had a hard time comparing their products with their competitors due to different naming conventions. Because of that, each of them had to define and maintain up to date a table mapping the vulnerabilities their product could detect vs. those detected by their competitors.

Tasks

- - - - -

A mailing list was created by Marc Dacier for the Group 5 participants to discuss the issues related to naming conventions (vns-public@zurich.ibm.com).

The vendors in the subgroup meeting (L-3, ISS, and Cisco) each offered to select 15 well-known vulnerabilities, name them according to the scheme, and present them to the mailing list for review. If a naming convention proved successful, the group would present its results to related standardization efforts, e.g. the IETF IDWG, and enlarge the group of contributors at that time.