Vulnerability Response Decision Assistance

Hal Burch[1], Art Manion[1], and Yurie Ito[2]

[1] CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 15213, United States
{hburch,amanion}@cert.org

[2] Japan Computer Emergency Response Team Coordination Center, Hirose Bld. 11F, 3-17 Kanda Nishikicho, Chiyoda-Ku, Tokyo, 101-0054, Japan
yito@jpcert.or.jp

Abstract: Each year, thousands of new software vulnerabilities are reported, and affected organizations must analyze them and decide how to respond. Many organizations employ ad hoc systems of decision making, which often result in inconsistent decisions that do not properly reflect the concerns of the organization at large. VRDA (Vulnerability Response Decision Assistance) allows organizations to leverage the analysis effort at other organizations and to structure decision-making. VRDA enables organizations to spend less time analyzing vulnerabilities in which they are not interested, to make decisions more consistently, and to structure their decision making to better align with the goals of the organization. VRDA consists of a data exchange format, a decision making model, a decision model creation technique, and a tool embodying these concepts. One response team is employing a basic form of VRDA to cut the number of vulnerabilities analyzed by a factor of two. Another response team is developing and testing a VRDA implementation within their organization.

Keywords: VRDA, KENGINE, vulnerability, decision support, decision tree, severity metric

1 Overview

In 2006, CERT/CC recorded more than 8,064 vulnerabilities [1] and NVD recorded 6,604 vulnerabilities [2]. For each vulnerability, organizations must analyze the vulnerability to determine which software systems are affected, the impact of a successful exploit, and how difficult it is for an attacker to successfully exploit the vulnerability. Once this analysis is complete, the organization must determine whether or not the vulnerability warrants further action, whether that be producing an alert, conducting further analysis, or otherwise responding to the vulnerability. This expensive analysis is repeated at organizations around the world, resulting in a large duplication of effort.

Once the analysis is complete, an organization must decide how they will respond to the vulnerability. Responses vary; it may ignore the issue, immediately starting the patching process, record the issue for the next periodic update, or publish the information about the vulnerabilities (internally or externally). The actions warranted

by a particular vulnerability are based on the number of systems affected, the value of those systems, how likely exploit is perceived to be, the impact of a successful exploit, and the risk and expense induced by testing the patch and deploying it.

In many organizations, the decision about which actions are warranted is made in an ad hoc manner based on staff experience. This results in incorrect and inconsistent decisions that tend to reflect the goals of the organization as perceived by the decision maker, not the true organizational goals. Multiple decision makers only confound the problem.

We propose a new system called VRDA (Vulnerability Response Decision Assistance). VRDA address these problems by giving structure to the decision making process. This structuring enables interchange of the analysis, reducing the duplication of effort. In addition, the concerns of the organization can be codified into a model, making the decisions more consistent and better aligned with the organizational goals. VRDA is designed to answer the questions about which vulnerabilities an organization should be responding to, what the response should be, and with what priority.

2 VRDA

VRDA is composed of multiple components: a break-down of facts (attributes or properties of vulnerabilities), a method for recording relationship between affected systems and fact values, a data exchange architecture, a data exchange format, a format for modeling decision, and a method for creating decision models. To maximize the value that organizations can derive from it, VRDA is designed to be open and modular. Thus, an organization need only use the components that best meet their needs. The various components of VRDA are explained throughout this paper. An overview of VRDA usage is shown in Figure 1.

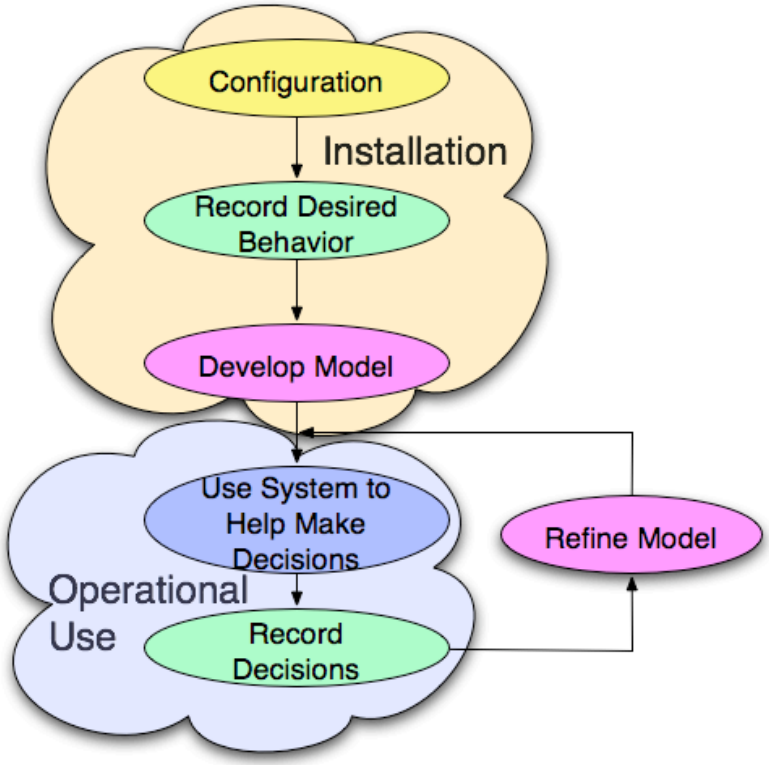

Figure 1: VRDA system overview

During the installation phase, the organization using VRDA must determine what information (facts) is necessary to make accurate response decisions (perform tasks). VRDA is configured with the selected facts and tasks. Facts may be obtained from an upstream provider; in otherwords, the organization may subscribe to a feed of VRDA facts from a computer security incident response team (CSIRT) that has vulnerability analysis capability. Analysts knowledgeable about the organization's IT assets and business operations must then train the system, recording the appropriate response (desired behavior) for a set of sample vulnerability reports. This process encodes the organization's values into a decision model.

With a working decision model, VRDA can enter the operational phase. As analysts score vulnerabilities, or append organization-specific data to the upstream feed, VRDA suggests appropriate responses. The actual response decision is also recorded, so that the model can be refined if necessary. For example, if VRDA regularly suggests a certain response, but analysts often disagree, then the model most likely does not accurately reflect the organization's values, or VRDA may not be configured with the necessary facts.

VRDA is designed to tolerate incomplete information and "best guesses" by analysts. VRDA often does not examine every fact to reach a decision, and even a few off-by-one scoring errors are unlikely to significantly affect the decision. A well-

constructed model will quickly and accurately enable easy decisions, such as ignoring vulnerability reports that do not affect the organization or flagging high severity reports.

2.1 Facts

In VRDA terms, *facts* are assertions about vulnerabilities. Facts can also be thought of as attributes, characteristics, or properties of vulnerabilities. The act of an analyst researching and recording facts is called *scoring*. A VRDA fact is not a "fact" in the strict definition, that is, VRDA facts are not indisputable and may be subject to interpretation. Facts represent the best information available to the analyst at the time of scoring, and fact values may be determined in part by analyst judgment and experience.

VRDA proposes a set of "core" facts. An organization may, however, create and score any additional facts that affect response decisions. A VRDA instance that does not consider significant facts will likely produce incorrect suggestions. In other words, the set of facts used by a VRDA instance is not fixed.

Most facts have an ordered range of possible values. To balance the value of accuracy with the cost of achieving accuracy, the granularity for fact values is fairly low. For example, a common range contains four possible values: "low," "low-medium," "medium-high," and "high." VRDA prefers four values to reduce the tendency of analysts to select the median "safe" value. When lacking sufficient information to make a confident selection, analysts should make an intuitive, educated guess.

VRDA facts are organized into three categories: vulnerability facts, world facts, and constituency facts. The distinctions between categories do not affect the decision modeling. The distinctions do help inform how the facts apply and whom should provide them.

Vulnerability facts apply directly to vulnerabilities. These facts are typically inherent technical properties of vulnerabilities. Vulnerability facts apply regardless of other world or constituency facts. For example, a denial-of-service impact exists whether or not exploit code is available or the vulnerable software is deployed within a given constituency. While anyone can score vulnerability facts, VRDA expects an experienced analyst (e.g., a CSIRT or response team) to score vulnerability facts and provide them to downstream consumers. The following is a working list of vulnerability facts. Further explanation is provided in VRDA documentation.

Security Product – Does the vulnerability affect a security product? (Yes/No)

Network Infrastructure Product – Does the vulnerability affect a network infrastructure product? (Yes/No)

Multiple Vendors – Does the vulnerability affect multiple vendors? (Yes/No)

Impact 1 – What is the general level of impact of the vulnerability on a system? (Low, Low-Medium, Medium-High, High)

Impact 2 – What are the levels of impact for confidentiality, integrity, and availability of the vulnerability on a system? (Low, Low-Medium, Medium-High, High)

Access Required – What access is required by an attacker to be able to exploit the vulnerability? (Routed, Non-routed, Local, Physical)

Authentication – What level of authentication is required by an attacker to be able to exploit the vulnerability? (None, Limited, Standard, Privileged)

Actions Required – What actions by non-attackers are required for an attacker to exploit the vulnerability? (None, Simple, Complex)

Technical Difficulty – What degree of technical difficulty does an attacker face in order to exploit the vulnerability? (Low, Low-Medium, Medium-High, High)

World facts apply to "meta" information about vulnerabilities. World facts describe states of the world or environment in which vulnerabilities exist. World facts apply to all constituencies. Anyone can score world facts, although VRDA expects experienced analysts to score and provide them to downstream consumers. The following is a working list of world facts. Further explanation is provided in VRDA documentation.

Public Attention – What amount of public attention is the vulnerability receiving? (None, Low, Low-Medium, Medium-High, High)

Quality of Public Information – What is the quality of public information available about the vulnerability? (Unacceptable, Acceptable, High)

Exploit Activity – What level of exploit or attack activity exists? (None, Exploit exists, Low activity, High activity).

Report Source – What person or group reported the vulnerability?

Constituency facts measure information about vulnerabilities that is specific to a given constituency. For example, a CSIRT may consider a large scope, like entire site, or even the entire Internet, when deciding how to respond. A system administrator may consider a smaller scope, such as a site, lab, or business unit. Constituency facts most likely differ between constituencies and must be provided by (or on behalf of) the constituency making the response decision.

Population – What is the population of vulnerable systems within the constituency? (None, Low, Low-Medium, Medium-High, High)

Population Importance – How important are the vulnerable systems within the constituency? (Low, Low-Medium, Medium-High, High)

Default Fact Sets (DFS) are preset fact values for a set of facts. DFS help analysts consistently score similar vulnerabilities. DFS was conceived to help score impact. For example, an analyst may determine that most vulnerabilities that allow an attacker to execute arbitrary code have the value "high" for the confidentiality, integrity, and availability *Impact 2* facts. The analyst could define a DFS named "execute arbitrary code" that sets the *Impact 2* facts when the DFS is applied to a vulnerability report. In addition to applying the default fact sets, the name of the DFS itself is a fact (i.e., that the analyst applied the DFS named "execute arbitrary

code" may factor into the response decision). In this sense, a DFS is similar to a keyword or tag.

2.2 Light-weight affected product tags

One barrier to the exchange of facts about vulnerabilities is that some important facts, such as population size and population importance, are constituency facts that cannot be determined by an outside entity. If the population of a particular affected system is high in one constituency, it does not guarantee the population is high in any subset of that constituency. For example, Microsoft Internet Explorer has a high population on the internet as a whole, but a given organization may have few systems using Microsoft Internet Explorer, either because they, by policy, use a different browser or because they have few Microsoft Windows systems.

As a result, constituency facts are generically not helpful to communicate. Light-weight affected product tags (LAPTs) communicate the set of affected systems so that the constituency facts can be computed. A vulnerability is tagged with the list of affected systems encoded as LAPTs. When VRDA receives a vulnerability with LAPTs, it consults a database to determine the constituency fact values for each LAPT listed and, for each constituency fact, selects the value corresponding to the worst (most likely to result in action) value appearing in the LAPTs. For example, if a vulnerability affects a high population LAPT and a low population LAPT, the population is set to high. The worst value is selected as the best guess and the least likely to hide an important vulnerability. Few vulnerabilities affect a large set of low population systems that together represent a high population.

This combination can lead to perhaps unexpected results, when a low population and high value product is affected along with a high population but low value product. The resulting constituency facts are a high population size and value, which is arguably incorrect. The root of the problem, however, is not LAPTs but rather simplification in the fact modeling. A population with a few high-value systems and a lot of low-value systems is difficult to model in terms of purely population size and population value. We believe that modeling as we have is more useful, but if an organization was so inclined, VRDA can be configured to population sizes for each population value, in which case this problem would not arise.

LAPTs are designed to be general descriptions of the affected system. Thus, it specifies the vendor and the system. Unless a population is large and many vulnerabilities affect only particular versions, the version is not included. For example, the Apache web server is a single LAPT. The motivations for this lack of specificity are as follows:

1. Difficulty of determining version information: It is often difficult to determine the exact versions affected, particular for commercial systems or systems that don't require patches to be applied in a particular order.
2. Cost of inventory maintenance: It is difficult enough to maintain inventory of the size of each software product. Keeping track of version numbers can make the problems an order of magnitude worse.

3. Limited usefulness: Most vulnerabilities discovered affect the current version of the affected software product, and often all recent versions. Thus, few vulnerabilities will be discarded by checking version numbers.

Most LAPTs refer to particular software products. When a vulnerability affects a technology or is an error common to many implementations of a technology, a "technology LAPT" can be utilized. For example, an error that is observed in SSL or in many SSL libraries might be attributed to the technology of "SSL", rather than listing all the affected products, although the list of LAPTs might also include products known to be affected. For common technologies, any list of products using the technology is doomed to be incomplete. Technology LAPTs provide a mechanism to describe a class of affected products to avoid this incompleteness.

2.3 Data Exchange

One of the goals of VRDA is to reduce the redundant analysis of vulnerabilities that takes place today. To this end, an organization must be able to obtain structured vulnerability information, in the form of fact values and LAPTs from another organization. Once an organization receives this information, they may refine the fact values or augment with additional fact values as desired and republish the results for use by other organizations. For example, a CSIRT with national responsibility might publish the information for a company's CSIRT who then passes information on to groups internal to that company. This second level might be in the form of alerts or a subset of the vulnerabilities in VRDA format, perhaps augmented with additional local information.

When an organization receives vulnerability information in the form of fact values and LAPTs, any previously unknown LAPTs must have their values recorded. Once they are recorded or if there are no new LAPTs, the constituency facts can be added to the vulnerability information. At this point, VRDA filters out based in its configuration. Ideally, VRDA would filter out the majority of the vulnerabilities, enabling an organization to focus its resources better. For vulnerabilities that pass this filter, the organization adds fact values for any local facts. Then, decisions are suggested based on the process described in "decision modeling."

The data exchange format is based on VULDEF [3]. Most of the optional fields of VULDEF, such as Solution, Related, and Exploit, are ignored by VRDA. The major differences from VULDEF to the VRDA exchange format is AffectedItem is augmented with a LAPT and FactList and facts tags are added to communicate the values of the facts.

An abbreviated example of the interchange format

```
<?xml version="1.0" encoding="UTF-8"?>
<Vulinfo>
  <VulinfoID>JVN#178394</VulinfoID>
  <VulinfoData>
    <Affected version="1.0" historyno="2">
      <AffectedItem lapt="Microsoft-Windows-XP"/>
      <AffectedItem lapt="Tech-HTTP-Server"/>
```

```
    </Affected>
    <FactList version="1.0" historyno="2">
      <ImpactConfidentiality>Low</ImpactConfidentiality>
      <ImpactAvailability>Low</ImpactAvailability>
      <ImpactIntegrity>Medium</ImpactIntegrity>
      <AccessRequired>Routed</AccessRequired>
...
</VulinfoData>
</Vulinfo>
```

2.4 Decision Modeling

VRDA makes decisions by modeling the process as a decision tree [4]. An example of a decision tree is shown in Figure 3. The evaluation of a decision tree begins at the root. At each node along the evaluation path, the child based on the attribute associated with that node. In the example decision tree, if the population is high, then the left child is followed, at which point the difficulty of exploit is considered. If the difficulty of exploit is low, then the decision tree evaluates to "must".

We selected decision trees because their behavior is clearly indicated and they can be hand-modified. Although we expect decision trees to be computed based on recorded decisions for past vulnerabilities, the computed trees might need refinement. For example, an organization may, as policy, not need independently verify reports from particular sources or may always respond to particular types of reports. Modifying a decision tree to represent such policies can be done clearly and simply. Alternative decision models, such as neural nets or linear combinations, might be able to better capture the intricacies of the decision making process, but it is difficult to understand what policy they implement and they lack the ability to predictably hand-tune the model.

We expect the resulting decisions to be imperfect. However, we compensate for that by giving gradients of decisions, rather than Boolean values. In particular, we use four levels: "must", "should", "might", and "won't". The goal is that the resulting decision level should not differ more than one from the "correct" value. Since, in our experience, experts often disagree more widely than that anyway, this accuracy may be ambitious. In any case, the decisions serve as a guideline, rather than a rule, to handlers. This allows for automated prioritization, including deciding to ignore vulnerabilities whose evaluation falls says "won't" for all decisions, which reduces the load on handlers within an organization

VRDA constructs decision trees using a similar algorithm to the one described in two references ([4] and [5]) which is based on a recursive selection of the attribute that reduces entropy the most. However, instead of pruning the decision tree after its construction, attributes that fail the chi-squared test of significance are not considered when constructing the tree.

Once the decision tree is constructor, the organization can modify the decision tree, change leaf values, changing the attribute used at a node, and have the algorithm compute a leaf value or an entire sub-tree. These operations allow the organization to codify an important and well-understood policy decision, such as always patching

certain types of vulnerabilities manually and then allowing the tree for the other vulnerabilities to be computed automatically.

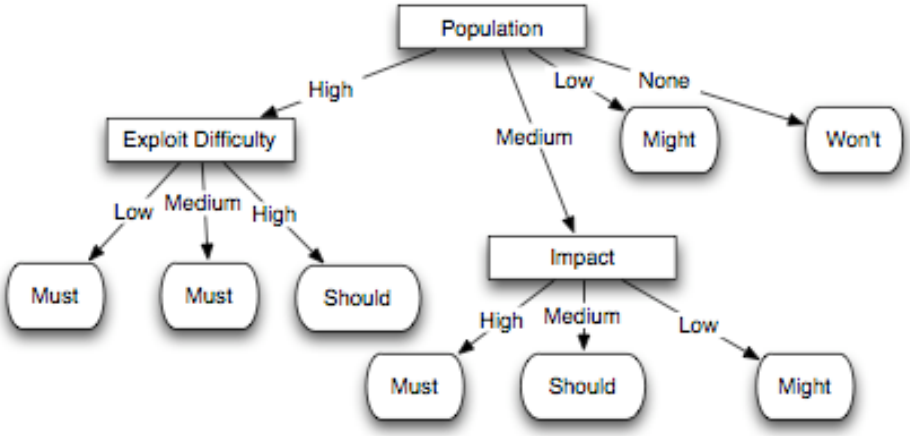

Figure 3: An example of a decision tree

3 Current Usage

CERT/CC uses a limited implementation of VRDA to decide which vulnerability reports require the attention of a human analyst. A brief study showed that two facts heavily influenced this decision: the population of affected systems and the broad impact of the vulnerability. By recording these two facts and applying a simple decision model, the number of reports assigned to analysts was reduced by half.

JPCERT/CC has developed a web-based VRDA implementation called KENGINE. JPCERT/CC is using KENGINE internally and is testing with several constituents. In addition to implementing the VRDA specification, KENGINE provides workflow management and reporting features to monitor performance and decision behavior. JPCERT/CC is planning to publish vulnerability information in VRDA format and to release KENGINE to the public.

4 Future Direction

The teams developing VRDA expect to do the following:

1. Score vulnerability reports and provide facts to downstream consumers, possibly the general public. Consumers may use the facts as they wish, with or without the decision support component.

2. Provide documentation and guidance for consumers to use the decision support component.
3. Deploy beta VRDA systems to knowledgeable consumers to gain real-world experience and determine if VRDA is useful and viable.
4. Refine VRDA according to results of the beta deployments and community feedback.
5. Examine interoperability with other vulnerability information systems, particularly the Common Vulnerability Scoring System (CVSS) [6].

5 Related Work

A number of efforts have been made (or are currently underway) to represent vulnerability information and provide consumers some means to determine severity, leading to an appropriate response. These efforts include metrics, information exchange formats and protocols, and vulnerability information databases. VRDA draws ideas and inspiration from many of these efforts while proposing a decision support approach to vulnerability response.

5.1 Common Vulnerability Scoring System (CVSS)

Perhaps the most similar and contemporary effort, the Common Vulnerability Scoring System (CVSS) "…is designed to rank information system vulnerabilities and provide the end user with a composite score representing the overall severity and risk the vulnerability presents." [6] VRDA is similar to CVSS in some ways, particularly in the representation of facts (VRDA) and metrics (CVSS) used to describe vulnerabilities. It may well be possible to use the decision support component of VRDA with CVSS metrics. While VRDA specifies a set of core facts, any fact that contributes significantly to a response decision for an organization can and should be considered. In contrast, CVSS specifies a fixed list of metrics.

A more notable difference is that VRDA uses decision support concepts to generate individual response decisions, while CVSS assigns fixed values to metrics and applies a single equation to calculate severity. VRDA effectively allows organizations to set their own individual values and make their own individual response decision. This design choice comes at a cost – VRDA requires more effort on the part of the organization than CVSS. And in fairness, CVSS does provide limited environmental metrics that modify the overall score based on characteristics that are unique to individual organizations.

Although CVSS and VRDA measure vulnerability characteristics similarly, the two systems are designed with somewhat different goals. CVSS aims to provide an overall severity score, while VRDA focuses on the decision-making aspect of how an individual organization responds to vulnerabilities.

5.2 Exchange Formats

There exist a variety of vulnerability information exchange formats, including the Common Announcement Interchange Format (CAIF) [7], the Common Model of System Information (CMSI) [8], the EISPP Common Advisory Format Description [9] and the Deutsches Advisory Format (DAF) [10]. These formats generally describe ways to exchange and present vulnerability information for use in advisory documents and include functionality that is unnecessary for VRDA. In some cases, the formats cannot be reasonably extended to meet VRDA requirements.

5.3 Other Work

Other related work includes (in no particular order): the Purdue University CERIAS Cassandra tool [11], the CERT Coordination Center/US-CERT Metric [12], MITRE Common Vulnerabilities and Exposures (CVE) [13] and Open Vulnerability and Assessment Language (OVAL) [14], the VULnerability Data publication and Exchange Format (VULDEF) [3] (used as the basis for the VRDA exchange format), NIST ICAT (now deprecated) [15], the National Vulnerability Database (NVD, successor to ICAT) [2], the, the Vulnerability and eXposure Markup Language (VuXML) [16], the Open Source Vulnerability Database (OSVDB) [17], and SIGVI [18].

References

1. CERT/CC Statistics 1988 – 2006, http://www.cert.org/stats/
2. National Vulnerability Database (NVD) Statistics, http://nvd.nist.gov/statistics.cfm
3. Terada, M.: VULDEF: The VULnerability Data publication and Exchange Format data model, http://jvnrss.ise.chuo-u.ac.jp/jtg/vuldef/index.en.html
4. Russell, S., Norvig P.: Artificial Intelligence: A Modern Approach. Prentice-Hall (1995)
5. Moore, A.: Decision Trees, http://www.autonlab.org/tutorials/dtree.html
6. Forum of Incident Response Teams: Common Vulnerability Scoring System (CVSS), http://www.first.org/cvss/, http://www.first.org/cvss/cvss-guide.html
7. RUS-CERT: Common Announcement Interchange Format (CAIF), http://www.caif.info/
8. Grobauer, Bernd: CVE, CME,..., CMSI? – Standardizing System Information, http://www.first.org/conference/2005/papers/dr.-bernd-grobauer-paper-1.pdf
9. European Information Security Promotion Programme (EISPP): Common Advisory Format Description 2.0, http://www.eispp.org/commonformat_2_0.pdf
10. Deutscher CERT-Verbund: Deutsches Advisory Format (DAF), http://www.cert-verbund.de/daf/index.html, 2004.
11. CERIAS Cassandra tool, https://cassandra.cerias.purdue.edu/main/index.html
12. US-CERT Vulnerability Notes Field Descriptions – Metric, http://www.kb.cert.org/vuls/html/fieldhelp#metric
13. Common Vulnerabilities and Exposures (CVE), http://cve.mitre.org/
14. Vulnerability and Assessment Language (OVAL), http://oval.mitre.org/
15. ICAT Metabase, http://icat.nist.gov/icat_documentation.htm, http://web.archive.org/web/20050320143644/http://icat.nist.gov/icat_documentation.htm

16. Vulnerability and eXposure Markup Language (VuXML), http://www.vuxml.org/
17. OSVDB: The Open Source Vulnerability Database, http://osvdb.org/
18. SIGVI, http://sigvi.sourceforge.net/what_is.php