A Decision Support System for Vulnerability Response

Hal Burch
CERT/CC

Art Manion
CERT/CC

Yurie Ito
JPCERT/CC

Abstract: Every day, CSIRTs respond to vulnerability reports, judging which are important enough to publish. For the thousands of reports received each year, CSIRTs must analyze the vulnerability to determine the software systems it affects, the actions an attacker must take to successfully exploit it, and the resulting impact. After analysis, CSIRTs too often formulate their response using an ad-hoc, informal decision-making process based solely on individual and organizational experience. Vulnerability Response Decision Assistance (VRDA) is designed to compile analysis efforts between organizations and to help rationally structure decisions. VRDA first allows organizations to tap into structured vulnerability information previously analyzed elsewhere. It then employs mathematical models to filter out irrelevant vulnerabilities and recommends products or actions to respond to the vulnerability. VRDA consists of a data exchange format, a decision making model, a decision model creation technique, and a tool embodying these concepts. VRDA enables organizations to spend less time analyzing immaterial vulnerabilities, to make decisions more consistently, and to structure decisions to better align with goals.

Overview

Each year, CSIRTs must process a large volume of vulnerability information. CERT/CC[1] recorded 8,064 vulnerabilities [CERT 06] and CVE recorded 6,604 vulnerabilities [NVD 06] in 2006. CSIRTs analyze each one to determine which software systems are affected, the degree of difficulty for an attacker to successfully exploit the vulnerability, and its impact. Once this analysis is complete, the CSIRT must determine whether or not the vulnerability warrants further action, whether that means producing an advisory, doing further analysis, or responding to the vulnerability in some other way. This expensive analysis is replicated at multiple CSIRTs around the world, resulting in considerable duplication of effort.

In addition to this redundant effort, many organizations make decisions about vulnerabilities in an ad hoc manner, usually meaning that resident experts use their experience to guide their decision-making. The experts may be able to explain the general guidelines they follow, but experts at the same organization can and often do disagree

[1] CERT and CERT Coordination Center are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

about what actions a vulnerability warrants. Moreover, the decisions of individuals at a CSIRT may not represent the policies of the CSIRT or the concerns of its constituency.

We propose the VRDA system to reduce duplication of effort, improve efficiency, and provide structure to decision-making. This system includes an interchange model, a model for computer-assisted decision-making, and a tool embodying those concepts. The system is designed so that a CSIRT need only employ the system features that meet their needs.

Tasks

Tasks, in this system, are defined as actions a VRDA user performs in response to a vulnerability report. Tasks are a major component of VRDA output – VRDA helps a user decide which tasks make up an appropriate response. Examples of tasks include publishing an advisory, notifying select groups, or initiating a patching process. One important task is to *intentionally* take no action, so as to ignore vulnerability reports that do not affect a CSIRT's constituency or are otherwise not severe enough to warrant further effort.

Facts

Facts, as defined here, are properties of vulnerabilities. A vulnerability report is represented in VRDA by a set of facts. VRDA proposes a set of core facts; however, any property of a vulnerability that informs the response decision can (and should) be recorded. Examples of core VRDA facts include impact, access and authentication requirements, exploit activity, and patch availability.

Lightweight affected product tags (LAPTs) are identifiers for products that are affected by vulnerabilities. LAPTs consist of LAPT names and sets of product-related facts. LAPT names are shared among all VRDA users, but LAPT facts are specific to individual users. A LAPT name is typically a vendor, product, or technology (e.g. *Microsoft-Window*s, *Apple-QuickTime*, or *ICMP*). Two examples of LAPT facts are the population and importance of affected systems. VRDA users provide values for LAPT facts, since users typically know their inventories and asset values better than CSIRTs. In practice, a CSIRT labels a vulnerability report with an LAPT and a VRDA user provides the values for population and importance associated with the LAPT name.

Default Fact Sets (DFS) are sets of facts with preset values. Similar vulnerabilities or classes of vulnerabilities have similar properties, and this should be reflected by VRDA fact values. DFS provides a way to define which facts and values are common to certain types of classes of vulnerabilities. Like LAPTs, DFS consist of a DFS name and one or more fact values. Unlike LAPTs, DFS facts are not tied to product identity and are not necessarily provided by the VRDA user. An analyst can use a DFS to quickly and accurately record facts that are generally applicable to a class of vulnerability reports.

DFS facts are not fixed, they are set when the DFS is applied and can be modified by an analyst as needed. The name of a DFS is also recorded as a fact. An example DFS is cross-site scripting (*XSS*). For an *XSS* vulnerability report, a DFS applies the default values for attacker access and impact facts, and then labels the report with the DFS name, *XSS*. An analyst can modify facts set by the DFS as necessary.

Data Exchange

Rather than attempting to enumerate all possible structured vulnerability information one may want to represent, the data exchange format is generic and flexible, allowing each organization to decide what data it wishes to publish and consume. Although there is a core set of facts, it may be extended or omitted to better serve the needs of organizations.

The format is based on VULDEF [VULDEF]. However, VRDA differs from VULDEF in that fields required by VULDEF that are not central to the exchange of structured information are optional fields in VRDA. The format consists of a label for the vulnerability, a title, a list of affected software, and a set of facts with values.

The process model we describe starts with one organization publishing structured vulnerability information using the VULDEF format. Other organizations can then use that information, combining it with their own analyses to produce an enhanced set of facts upon which they will base their decisions. After their additions and refinements, the organization can then share the structured information further.

For example, a CSIRT at a large company might download information published that day by the CERT/CC. It can then cross-reference the list of affected software with its inventory information to determine which internal systems are affected. Based on that evaluation, it can determine which business units need to be notified. It republishes to those units, which may then, in turn, employ a similar process to make decisions.

Model for Decision Making

The decision making model is based on decision trees. (An example of a decision tree is shown in Figure 1.) The evaluation of a decision tree begins at the root. At each node along the evaluation path, the path follows the "child" corresponding to the value of the attribute associated with that node. In the example decision tree, if the population is high, then you follow the left child, at which point the difficulty of exploit is considered. If the difficulty of exploit is low, then the decision tree points to "must."

We used decision trees because recommended behaviors are clearly indicated and can be easily modified. Although we expect the decision trees to be computed based on recorded decisions for past vulnerabilities, these computed trees might need to be refined.

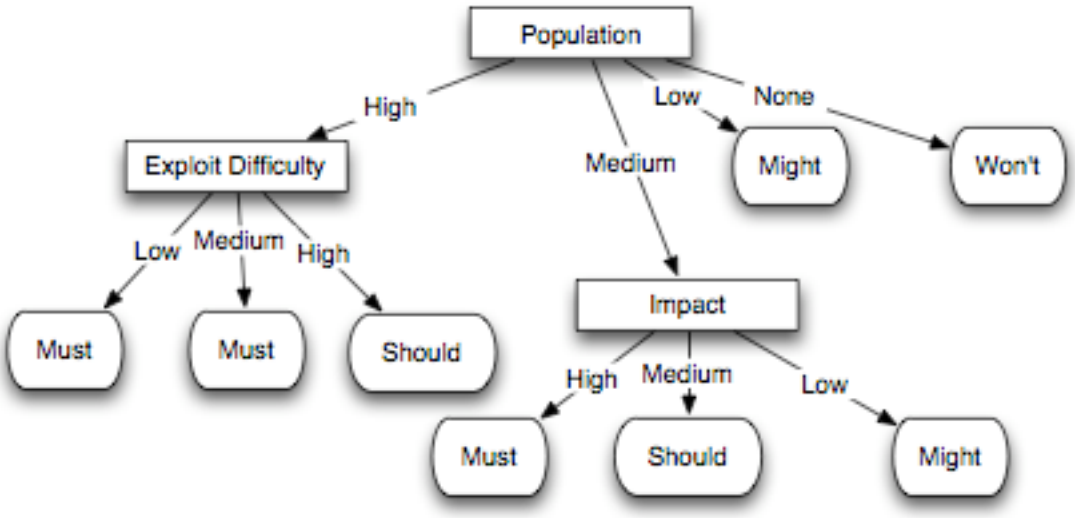

Figure 1 - Sample Decision Tree for VRDA

For example, an organization may not, as a matter of policy, need to independently verify reports from a certain source. Or, it may always respond in the same way to other types of reports. In a decision tree, modifying the tree to represent these policies can be done clearly and simply. Alternative decision models, such as neural nets or regression, might be better able to capture the intricacies of the decision making process, but these models lack any ability to convey what policy they implement other than by example. They also do not provide the ability to simply "handtune" 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, for vulnerability handlers. This allows for automated prioritization, including deciding to ignore vulnerabilities whose evaluation is *won't* for all decisions. This reduces the load on handlers within a CSIRT.

Future Direction

The CERT/CC employs a basic form of this system. Using this decision model, slightly more than half of the vulnerabilities recorded by our public monitoring team are *not* assigned to a vulnerability analyst. This reduces the workload, both perceived and actual, of the vulnerability analysts, improves morale, and makes them more efficient. Of the thousands of vulnerabilities discarded thus far, less than five vulnerabilities were later found to warrant assignment. Each of these errors resulted from a public monitor not being aware of the prevalence of the software product.

That is, the decision-making was wrong because the input was wrong. To reduce such errors, the CERT/CC is developing an independent "Ubiquity" system to better estimate populations of software products.

JPCERT/CC[2] developed a tool called "KENGINE" that implements the concepts described here. KENGINE manages vulnerability handling decision models, includes analysis parameters, values, decision rules, LAPTs, and vulnerability handling workflow. KENGINE helps JPCERT/CC vulnerability analysts make more consistent and effective vulnerability response decisions. KENGINE includes the ability to import data from external sources, record facts, cross-reference inventory information with lists of systems affected, decision making, decision recording, an interface to the task tracking workflow system, and model development. KENGINE can also produce statistical reports, such as vulnerability handling progress, handling workload, decision trends, decision review, and deviations between actual actions and VRDA suggestions. JPCERT/CC plans to release KENGINE to the public.

References

[CERT 06] CERT Coordination Center, *CERT/CC Statistics 1988 – 2006*, <http://www.cert.org/stats/>

[NVD 06] National Vulnerability Database, *National Vulnerability Database Statistics*, <http://nvd.nist.gov/statistics.cfm>

[VULDEF] Terada, Masato, *VULDEF: Security Advisory Publication Format Data Model*, <http://jvnrss.ise.chuo-u.ac.jp/jtg/vuldef/index.en.html>

[2] Japan Computer Emergency Response Team Coordination Center

