Prioritizing Vulnerability Response

Jonathan M Spring, Eric Hatleback, Allen Householder, Art Manion

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM20-0023

Motivation

We want an evidence-based vulnerability management system

CVSS (Common Vulnerability Scoring System) is:

- Limited to technical severity
- Has some design inconsistencies (See *Towards Improving CVSS*)

We propose a Stakeholder-Specific Vulnerability Categorization (SSVC) as an improvement

- Focus is on *decisions*, not technical severity
- Transparent, role-specific recommendations
- Experiment design to test process consistency

SSVC contributions

1. Decision process and descriptions that could be used to make vulnerability management decisions
2. Method for how a justifiable decision process and descriptions can be constructed, adapted, and tested

(1) is valuable, but it is a proposal, not a final answer

(2) is perhaps more important because it lets you adapt

This talk will summarize the content (1) and then our method (2)

SSVC roles

Propose different decision-making for different roles:

- Patch developer
- Patch applier or deployer
- Coordinator

These track the roles in coordinated vulnerability disclosure

This contrasts with CVSS, which is often used as a one-size-fits-all

For each, a decision is the priority of action on a work item

- A work item is (generally) to develop or deploy a patch

SSVC priorities

For each role, a decision is the priority of action on a work item

- Propose 4 levels of priority:
  - Defer (lowest)
  - Scheduled
  - Out-of-band
  - Immediate (highest)

The risk due to vulnerabilities in each priority class are equivalent

A user might prioritize within each priority class based on other considerations (ease, cost to fix, etc.)

Priorities and CVSSv3.1

CVSS suggests the following priority categories:

- None – 0.0
- Low – 0.1 to 3.9
- Medium – 4.0 to 6.9
- High – 7.0 to 8.9
- Critical – 9.0 to 10.0

But CVSS is only designed to be accurate ± 0.5

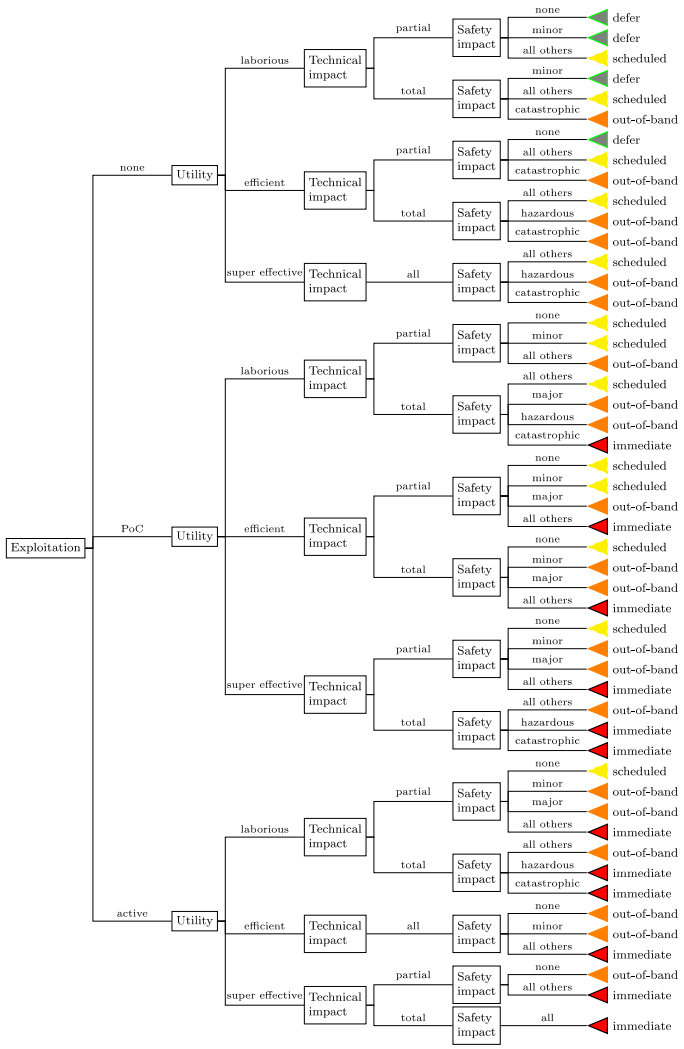
And it seems like it's actually accurate ± 1.5 [1]. Thus:

- 8.5 and 9.5 are equivalent by design (CVSS categories are not actually separate)
- 4.0 and 6.9, that is 5.5 ± 1.5, are functionally equivalent (CVSS medium is actually just one qualitative category anyway)

[1] Allodi L.; Cremonini M.; Massacci F.; & Shim W. *The Effect of Security Education and Expertise on Security Assessments: The Case of Software Vulnerabilities*. In WEIS 2018, Figure 1.

Deciding priorities

Given about 4 decision points, for each role, we made a decision tree to capture the options and outcomes



Decision points

The different roles have some different decision points

Patch Deployer:

- Exploitation
- Utility
- Technical Impact
- Safety Impact

Patch Applier:

- Exploitation
- Exposure
- Mission Impact
- Safety impact

All six of these decision points have detailed descriptions

Example decision point description

Exploitation (Evidence of Exploitation of a Vulnerability)

Measures the present state of exploitation of the vulnerability. Our intent is not to predict future exploitation but only to acknowledge the current state of affairs. Predictive systems, such as EPSS, could be used to augment this decision or to notify stakeholders of likely changes

None	There is no evidence of active exploitation and no public proof of concept (PoC) of how to exploit the vulnerability.
PoC (Proof of Concept)	One of the following cases is true: (1) private evidence of exploitation is attested but not shared; (2) widespread hearsay attests to exploitation; (3) typical public PoC in places such as Metasploit or ExploitDB; or (4) the vulnerability has a well-known method of exploitation. Some examples of condition (4) are open-source web proxies serve as the PoC code for how to exploit any vulnerability in the vein of improper validation of TLS certificates. As another example, Wireshark serves as a PoC for packet replay attacks on ethernet or WiFi networks.
Active	Shared, observable, reliable evidence that the exploit is being used in the wild by real attackers; there is credible public reporting.

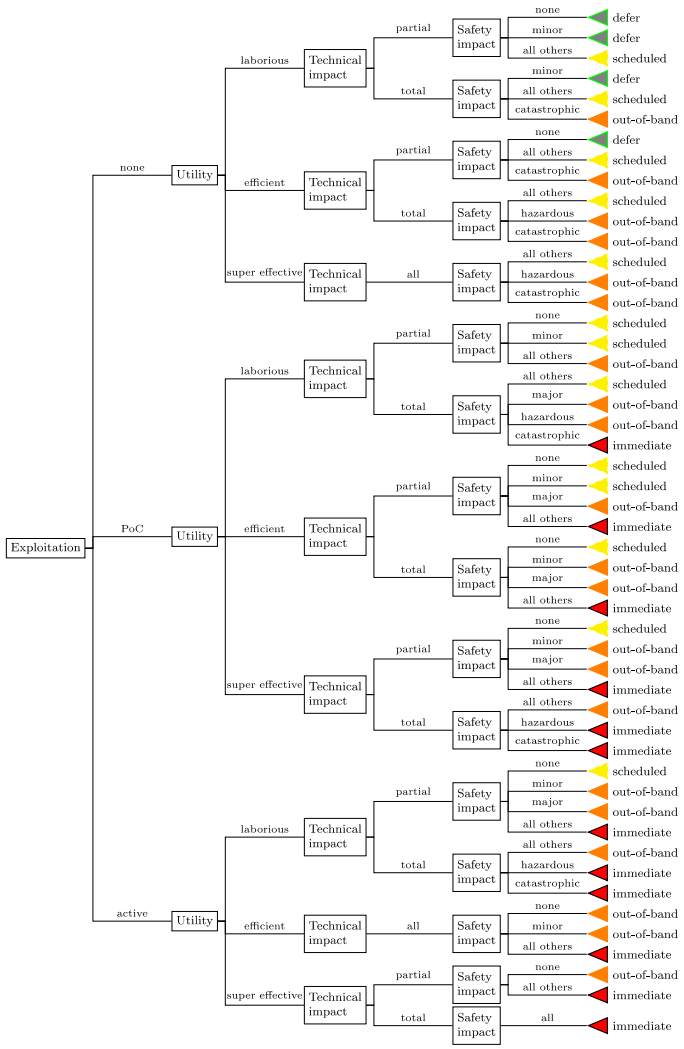
Decision trees

A decision tree is a convenient representation of logic statements

- Easy to automate, given the four data inputs

We assume humans will evaluate the decision points, though

- Many decisions are reusable. For example, exposure applies to the system, not the vulnerability
- Exploitation = PoC may be definable as a clever internet search (TBD)



Evaluation and testing

We conducted several rounds of internal testing before publishing

1. Informal evaluations of these trees by selecting a past CVE and discussing how each author would evaluate the priority of that vulnerability
   - About 8 rounds of this
   - Results: refined decision points, better defined the scope, clarified what it means for a vul to be in a system, and combined some draft trees
2. Formal pilot study to measure inter-rater agreement
   - Six analysts used the draft system to rate 9 sample vuls as both applier and developer, given a fixed description of the vul and the environment

Pilot study details

For each analyst, for each vulnerability, for each stakeholder group, do the following:

1. Start at the root note of the relevant decision tree (patch applier or patch developer)
2. Document the decision branch that matches the vulnerability for this stakeholder context
3. Document the evidence that supports that decision
4. Repeat this decision-and-evidence process until the analyst reaches a leaf node in the tree

Therefore, we measure how the analysts' decisions agree for each decision point in the tree, as well as the resulting priority

Pilot study results

What we want to measure is how consistently SSVC could be applied

Mixed results

For Fleiss' κ: <0 is bad, closer to 1 is better, 1 is perfect agreement

In response, we got rid of "portfolio value" and improved or clarified the other elements

	Safety Impact	Exploitation	Technical Impact	~~Portfolio Value~~	Mission Impact	Exposure	Dev Result	Applier Result
Fleiss' κ	0.116	0.807	0.679	0.257	0.146	0.480	0.226	0.295

We want your input!

We published SSVC because it now needs community input

(Not because we think it's finished)

This sort of inter-rater agreement study should be repeated with more diverse participants to further improve SSVC

Other rating systems, such as CVSS, could also benefit from this method for empirical testing of their scoring reliability

Summary

SSVC structures evidence-based decisions about the priority of vulnerabilities

Decision trees document and structure how to combine responses to decision points (such as exposure, exploitation, or utility) to reach priority decisions

Decisions are qualitative and remain qualitative in order to be transparent, reliable, and explainable

SSVC supports multiple stakeholder groups and we encourage people to tailor the details (with appropriate testing) to meet their needs or situation

SSVC provides two decision trees, for developers and appliers, as starting points or informed hypotheses

Thanks! Questions?

The SSVC paper is available:

https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=636379

Searching for “SSVC prioritizing” should work, too