Carnegie Mellon University
Software Engineering Institute

PRIORITIZING VULNERABILITY RESPONSE: A STAKEHOLDER-SPECIFIC VULNERABILITY CATEGORIZATION (VERSION 2.0)

Jonathan M. Spring
Allen Householder
Eric Hatleback
Art Manion
Madison Oliver
Vijay Sarvapalli
Laurie Tyzenhaus
Charles Yarbrough

April 2021

Contents

Introduction

This document defines a testable Stakeholder-Specific Vulnerability Categorization (SSVC) for prioritizing actions during vulnerability management. The stakeholders in vulnerability management are diverse. This diversity must be accommodated in the main functionality, rather than squeezed into hard-to-use optional features. Given this, we aim to avoid one-size-fits-all solutions as much as it is practical.

We will improve vulnerability management by framing decisions better. The modeling framework determines what output types are possible, identifies the inputs, determines the aspects of vulnerability management that are in scope, defines the aspects of context that are incorporated, describes how the model handles context and different roles, and determines what those roles should be. As such, the modeling framework is important but difficult to pin down. We approach this problem as a satisficing process. We do not seek optimal formalisms, but an adequate formalism.

Our decision-making process is based on decision trees. A decision tree represents important elements of a decision, possible decision values, and possible outcomes. We suggest decision trees as an adequate formalism for practical, widespread advice about vulnerability prioritization. We do not claim this approach is the only viable option. We suggest that specific vulnerability management stakeholder communities use decision trees. These suggestions are hypotheses for viable replacements for CVSS in those communities, but the hypotheses require empirical testing before they can be justifiably considered fit for use. We propose a methodology for such testing.

This document is version 2 of SSVC. The main improvements from version 1 are the addition of the coordinator stakeholder perspective, improvements to terminology, integration of feedback on decision point definitions, and tools to support practical use. These changes are described in more detail in Version 2 Changelog.

The document is organized as follows.

- Current State of Practice summarizes the current state of vulnerability management.
- Representing Information for Decisions About Vulnerabilities describes our design goals for an improved prioritization method.
- Vulnerability Management Decisions defines who the decision makers are and what options they are deciding among.
- Likely Decision Points and Relevant Data proposes a definition of decision points that a stakeholder might use during vulnerability management.
- Prioritization combines these decision points into example decision trees that can be used to prioritize action on a work item.

- Evaluation of the Draft Trees describes an early test of this method against the design goals, as much to show an adequate usability test methodology as for the results.
- Worked example provides examples of applying the methodology to sample vulnerabilities and discusses the relationship between SSVC and other vulnerability management prioritization systems.
- Future Work identifies ideas we haven't had time to incorporate yet.
- Limitations identifies limitations in the design.
- Conclusion provides some final thoughts.

Current state of practice

Vulnerability management covers "the discovery, analysis, and handling of new or reported security vulnerabilities in information systems [and] the detection of and response to known vulnerabilities in order to prevent them from being exploited" (Benetis et al. 2019). Prioritization of organizational and analyst resources is an important precursor to vulnerability analysis, handling, and response. The general problem is: given limited resources, which vulnerabilities should be processed and which can be ignored for now. We approach this problem from a pragmatic, practitioner-centered perspective.

The de facto standard prioritization language is CVSS (Spring and Illari 2019). CVSS avoids discussing decisions and, instead, takes **technical severity** as its fundamental operating principle. However, the standard does not provide clear advice about how CVSS scores might inform decisions (Wiles and Dugal 2019). SSVC instead considers technical severity as one decision point in vulnerability management. Severity should only be a part of vulnerability response prioritization (See, e.g., Farris et al. 2018).

Any re-adaptation of the basic CVSS mindset inherits its deeper issues. For example, arguments for the CVSS scoring algorithm have not been transparent and the standardization group has not justified the use of the formula either formally or empirically (Spring et al. 2018). One complaint is that a high CVSS score does not predict which vulnerabilities will be commonly exploited or have exploits publicly released (Allodi and Massacci 2012). Studies of consistency in CVSS scoring indicate that analysts do not consistently interpret the elements of a CVSS v3.0 score (Allodi et al. 2018). Because many adaptations of CVSS simply add additional metrics, we expect they will inherit such inconsistency. Analyst usability has so far been an afterthought, but we know from other areas of information security that usability is not well-served as an afterthought (Garfinkel and Lipford 2014). SSVC aims to learn from and improve upon these issues.

Surveys of security metrics (Pendleton et al. 2016) and information sharing in cybersecurity (Laube and Böhme 2017) do not indicate any major efforts to conduct a wholesale rethinking of vulnerability prioritization. The surveys indicate some options a prioritization method might consider, such as exploit availability or system attack surface. Representing Information for Decisions About Vulnerabilities describes our design goals for a pragmatic prioritization methodology that can improve and build on the state of current practice.

The target audience for SSVC is vulnerability managers of any kind. SSVC assumes that the vulnerability manager has identified that there is a vulnerability. We take our definition of **vulnerability** from (Householder, Wassermann, et al. 2020): "a set of conditions or behaviors that allows the violation of an explicit or implicit security policy." A variety of problems or issues with computer systems are important but are not vulnerabilities. SSVC presents a risk prioritization method that might be useful or at least allied to other risk management methods for these other kinds of issues. However, for this work we focus on decisions in the situation where there is a vulnerability and the vulnerability management team wants to decide what to do about it.

Representing Information for Decisions About Vulnerabilities

We propose that decisions about vulnerabilities—rather than their severity—are a more useful approach. Our design goals for the decision-making process are to clearly define whose decisions are involved; properly use evidentiary categories; be based on reliably available evidence; be transparent; and be explainable. Our inspiration and justification for these design goals are that they are the features of a satisfactory scientific enterprise (Spring, Moore, and Pym 2017) adapted to the vulnerability management problem.

To consider decisions about managing the vulnerability rather than just its technical severity, one must be clear about whose decisions are involved. Organizations that produce patches and fix software clearly have different decisions to make than those that deploy patches or other security mitigations. For example, organizations in the aviation industry have different priorities than organizations that make word processors. These differences indicate a requirement: any formalism must adequately capture the different decisions and priorities exhibited by different groups of stakeholders. As a usability requirement, the number of stakeholder groups needs to be small enough to be manageable, both by those issuing scores and those seeking them.

The goal of adequacy is more appropriate than optimality. Our search process need not be exhaustive; we are satisficing rather than optimizing (Simon 1996). Finding any system that meets all of the desired criteria is enough.

Decisions are not numbers. They are qualitative actions that an organization can take. In many cases, numerical values can be directly converted to qualitative decisions. For example, if your child's temperature is 105°F (40.5°C), you decide to go to the hospital immediately. Conversion from numerical to qualitative values can be complicated by measurement uncertainty and the design of the metrics. For example, CVSS scores were designed to be accurate with +/- 0.5 points of the given score (CVSS SIG 2019, sec. 7.5). Therefore, under a Gaussian error distribution, 8.9 is really 60% high and 40% critical since the recommended dividing line is 9.0. SSVC decisions should be distinct and crisp, without such statistical overlaps.

We avoid numerical representations for either inputs or outputs of a vulnerability management decision process. Quantified metrics are more useful when (1) data for decision making is available, and (2) the stakeholders agree on how to measure. Vulnerability management does not yet meet either criterion. Furthermore, it is not clear to what extent measurements about a vulnerability can be informative about other vulnerabilities. Each vulnerability has a potentially unique relationship to the socio-technical system in which it exists, including the Internet. The context of the vulnerability, and the systems it impacts, are inextricably linked to managing it. Temporal and environmental considerations should be primary, not optional as they are in CVSS.

We make the deliberation process as clear as is practical; therefore, we risk belaboring some points to ensure our assumptions and reasoning are explicit. Transparency should improve trust in the results.

Finally, any result of a decision-making process should be **explainable** Explainable is defined and used with its common meaning, not as it is used in the research area of explainable artificial intelligence. An explanation should make the process intelligible to an interested, competent, non-expert person. There are at least two reasons common explainability is important: (1) for troubleshooting and error correction and (2) for justifying proposed decisions.

To summarize, the following are our design goals for a vulnerability management process:

- Outputs are decisions.
- Pluralistic recommendations are made among a manageable number of stakeholder groups.
- Inputs are qualitative.
- Outputs are qualitative, and there are no (unjustified) shifts to quantitative calculations.
- Process justification is transparent.
- Results are explainable.

Formalization Options

This section briefly surveys the available formalization options against the six design goals described above. Table 1 summarizes the results. This survey is opportunistic; it is based on conversations with several experts and our professional experience. The search process leaves open the possibility of missing a better option. However, at the moment, we are searching for a satisfactory formalism, rather than an optimal one. We focus on highlighting why some common options or suggestions do not meet the above design goals. We argue that decision trees are a satisfactory formalism.

We rule out many quantitative options, such as anything involving statistical regression techniques or Bayesian belief propagation. Most machine learning (ML) algorithms are also not suitable because they are both unexplainable (in the common sense meaning) and quantitative. Random forest algorithms may appear in scope since each individual decision tree can be traced and the decisions explained (Russell and Norvig 2011). However, they are not transparent enough to simply know how the available decision trees are created or mutated and why a certain set of them works better. In any case, random forests are necessary only when decision trees get too complicated for humans to manage. We demonstrate below that in vulnerability management, useful decision trees are small enough for humans to manage.

Logics are generally better suited for capturing qualitative decisions. Boolean first-order logic is the "usual" logic—with material implication (if/then), negation, existential quantification, and predicates. For example, in program verification, satisfiability problem (SAT) and satisfiability modulo theories (SMT) solvers are used to automate decisions about when some condition holds or whether software contains a certain kind of flaw. While the explanations provided by logical tools are accessible to experts, non-experts may struggle. Under special conditions, logical formulae representing decisions about categorization based on exclusive-or conditions can be more compactly and intelligibly represented as a decision tree.

Decision trees are used differently in operations research than in ML. In ML, decision trees are used as a predictive model to classify a target variable based on dependent variables. In operations research and decision analysis, a decision tree is a tool that is used to document a human process. In decision analysis, analysts "frequently use specialized tools, such as decision tree techniques, to evaluate uncertain situations" (Howard and Matheson 1983, viii). We use decision trees in the tradition of decision analysis, not ML.

Table 1: How Vulnerability Prioritization Options Meet the Design Goals

	Outputs are Decisions	**Pluralistic**	**Qualitative Inputs**	**Qualitative Outputs**	**Transparent**	**Explainable**
Parametric Regression	✗	✗	✓	✗	✗	✓
CVSS v3.0	✗	✗	✓	✗	✗	✗
Bayesian Belief Networks	✗	Maybe	✗	✗	✓	✓
Neural Networks	✗	✗	✗	✗	✗	✗
Random Forest	✓	✓	✓	Maybe	✗	Maybe
Other ML	✗	Maybe	✗	✗	✗	✗
Boolean First Order Logics	Maybe	Maybe	✓	✓	✓	Maybe
Decision Trees	✓	✓	✓	✓	✓	✓

Decision Trees

A decision tree is an acyclic structure where nodes represent aspects of the decision or relevant properties and branches represent possible options for each aspect or property. Each decision point can have two or more options.

Decision trees can be used to meet all of the design goals, even plural recommendations and transparent tree-construction processes. Decision trees support plural recommendations because a separate tree can represent each stakeholder group. The opportunity for transparency surfaces immediately: any deviation among the decision trees for different stakeholder groups should have a documented reason—supported by public evidence when possible—for the deviation. Transparency may be difficult to achieve, since each node in the tree and each of the values need to be explained and justified, but this cost is paid infrequently.

There has been limited but positive use of decision trees in vulnerability management. For example, Vulnerability Response Decision Assistance (VRDA) studies how to make decisions about how to respond to vulnerability reports (Manion et al. 2009). This paper continues roughly in the vein of such work to construct multiple decision trees for prioritization within the vulnerability management process.

Representation choices

A decision tree can represent the same content in different ways. Since a decision tree is a representation of logical relationships between qualitative variables, the equivalent content can be represented in other formats as well. The R package data.tree has a variety of both internal representations and visualizations.

For data input, we elected to keep SSVC simpler than R, and just use a CSV (or other fixed-delimiter separated file) as canonical data input. All visualizations of a tree should be built from a canonical CSV that defines the decisions for that stakeholder. Examples are located in SSVC/data. An interoperable CSV format is also flexible enough to support a variety of uses. Every situation in SSVC is defined by the values for each decision point and the priority label (outcome) for that situation (as defined in Likely Decision Points and Relevant Data). A CSV will typically be 30-100 rows that each look something like:

```
2,none,slow,diffuse,laborious,partial,minor,defer
```

Where "2" is the row number, *none* through *minor* are values for decision points, and *defer* is a priority label or outcome. Different stakeholders will have different decision points (and so different options for values) and different outcomes, but this is the basic shape of a CSV file to define SSVC stakeholder decisions.

The tree visualization options are more diverse. We provide an example format, and codified it in src/SSVC_csv-to-latex.py. Why have we gone to this trouble when (for example) the R data.tree package has a handy print-to-ASCII function? Because this function produces output like the following:

```
1    start
2     ¦--AV:N
3     ¦   ¦--AC:L
4     ¦   ¦   ¦--PR:N
...
31    ¦   ¦   ¦   ¦   ¦   ¦       ¦--A:L    Medium
32    ¦   ¦   ¦   ¦   ¦   ¦       °--A:N    Medium
33    ¦   ¦   ¦   ¦   ¦   °--C:N
34    ¦   ¦   ¦   ¦   ¦       ¦--I:H
35    ¦   ¦   ¦   ¦   ¦       ¦   ¦--A:H  Critical
```

This sample is a snippet of the CVSSv3.0 base scoring algorithm represented as a decision tree. The full tree can be found in doc/graphics/cvss_tree_severity-score.txt. This tree representation is functional, but not as flexible or aesthetic as might be hoped. The visualizations provided by R are geared towards analysis of decision trees in a random forest ML model, rather than operations-research type trees.

Vulnerability Management Decisions

This section will define our audience for decision advice and how we are scoping our advice on vulnerability management decisions. Viable decision guidance for vulnerability management should (at a minimum) consider the stakeholder groups, their potential decision outcomes, and the data needed for relevant decision points. This section covers the first of these three parts, and the following sections address the other parts in turn. The "who" is primarily about categories of stakeholders—suppliers, deployers, and coordinators—as well as their individual risk tolerances. The "what" is about the scope, both in how the affected system is defined and how much of the world an analyst should consider when estimating effects of a vulnerability.

While we strive to make our examples realistic, we invite the community to engage and conduct empirical assessments to test them. The following construction should be treated as an informed hypothesis rather than a conclusion.

Enumerating Stakeholders

Stakeholders in vulnerability management can be identified within multiple independent axes. For example, they can be identified by their responsibility: whether the group *supplies*, *deploys*, or *coordinates* remediation actions. Depending what task a team is performing in a supply chain, the team may be considered a supplier, deployer, or a coordinator. Therefore, one organization may have teams that take on different roles. For example, an organization that develops and uses its own software might delegate the supplier role to its development team and the deployer role to its IT operations team. On the other hand, organizations using a DevOps approach to providing services might have a single group responsible for both the supplier and deployer roles. Organizations may also be distinguished by the type of industry sector. While it might be useful to enumerate all the sectors of the economy, we propose to draft decision points that include those from multiple important sectors. For example, we have safety-related questions in the decision path for all suppliers and deployers. The decision will be assessed whether or not the stakeholder is in a safety-critical sector.

The choice not to segregate the decisions by sector is reinforced by the fact that any given software system might be used by different sectors. It is less likely that one organization has multiple responsibilities within the vulnerability management process. Even if there is overlap within an organization, the two responsibilities are often located in distinct business units with distinct decision-making processes. We can treat the responsibilities as non-overlapping, and, therefore, we can build two decision trees—one for each of the "patch supplier" and "patch deployer" responsibilities in the vulnerability management process. We leave "coordinating patches" as future work. Each of these trees will have different decision points that they take to arrive at a decision about a given unit of work.

The next two sections describe the decision space and the relevant decision points that we propose for these two responsibilities within the vulnerability management process.

The target audience for this paper is professional staff responsible for making decisions about information systems. This audience encompasses a broad class of professionals, including suppliers, system maintainers, and administrators of many types. It also includes other roles, such as risk managers, technical managers, and incident responders. Although every layperson who owns a computing device makes decisions about managing it, they are not the target audience. The following decision system may help such laypeople, but we do not intend it to be used by that audience.

While C-level executives and public policy professionals often make, shape, or incentivize decisions about managing information systems, they are not the target audience, either. To the extent that decision trees for vulnerability management help higher level policy decisions, we believe the best way to help policy makers is by making technical decisions more transparent and explainable. Policy makers may see indirect benefits, but they are not our primary audience and we are not designing an approach for them directly.

Enumerating Decisions

Stakeholders with different responsibilities in vulnerability management have very different decisions to make. This section focuses on the differences among organizations based on their vulnerability management responsibilities. Some decision makers may have different responsibilities in relation to different software. For example, an Android app developer is a developer of the app, but is a deployer for any changes to the Android OS API. This situation is true for libraries in general. A web browser developer makes decisions about applying patches to DNS lookup libraries and transport layer security (TLS) libraries. A video game developer makes decisions about applying patches released to the Unreal Engine. A medical device developer makes decisions about applying patches to the Linux kernel. The list goes on. Alternatively, one might view applying patches as including some development and distribution of the updated product. Or one might take the converse view, that development includes updating libraries. Either way, in each of these examples (mobile device apps, web browsers, video games, medical devices), we recommend that the professionals making genuine decisions do three things: (1) identify the decisions explicitly, (2) describe how they view their role(s), and (3) identify which software projects their decision relates to. If their decisions are explicit, then the decision makers can use the recommendations from this document that are relevant to them.

Enumerating Vulnerability Management Actions

SSVC models the decision of "With what priority should the organization take action on a given vulnerability management work unit?" to be agnostic to whether or not a patch is available. A unit of work means either remediating the vulnerability—such as applying a patch—or deploying a mitigation. Both remediation and mitigations often address multiple identified vulnerabilities.

The unit of work may be different for different stakeholders. The units of work can also change as the vulnerability response progresses through a stakeholder's process. We elucidate these ideas with the following examples.

Supplier Units of Work

On the input side of the Supplier process, Suppliers typically receive reports of vulnerabilities in one or more versions of their product. Part of the Supplier's task on initial report intake is to resolve the initial report into a set of products and versions that are affected by the reported vulnerability.

Our working assumption is that for SSVC purposes, the supplier's unit of work is the combination of the vulnerability with each affected product. This implies the need for Suppliers to be able to resolve whatever they receive to that level of granularity in order to make best use of SSVC.

Products will often need to be addressed individually because they may have diverse development processes or usage scenarios. There are a variety of ways a Supplier might need to resolve the set of affected products. For example, they might

- recognize, on further investigation of the initial report, that additional versions of the product are affected
- discover that other products are affected due to code sharing or programmer error consistent across products
- receive reports of vulnerabilities in third party libraries they utilize in one or more of their products
- receive fix bundles for third party libraries used in one or more of their products (where a fix bundle might resolve multiple vulnerabilities or add new features)

Without belaboring the point, the above methods are similar to how CVE Numbering Authorities discern "independently fixable vulnerabilities" (CVE Board 2020). We also note that SBOM(Jump and Manion 2019) seems well-placed to aid in that resolution process for the third-party library scenarios.

In the end, Suppliers provide remediations and/or mitigations for affected products. A supplier-provided remediation is usually a software update which contains fixes for multiple vulnerabilities and, often, new or improved features. Supplier output is relevant because it will become input to Deployers. SSVC focuses only on the remediation in this case; a set of remediations for multiple vulnerabilities is a fix bundle. Suppliers may also produce mitigations, such as recommended configuration changes, to limit the impact of a vulnerability.

Deployer Units of Work

Deployers are usually in the position of receiving remediations or mitigations from their Suppliers for products they have deployed. They must then decide whether to deploy the remediation or mitigation to a particular instance (or not). Whether they have the option of deploying only part of a remediation such as a fix bundle depends on whether the Supplier has engineered their release process to permit that degree of flexibility. For example, if service packs are fix bundles, the Supplier might choose to release individually deployable fixes as well.

The vulnerability management process for deployers has at its core the collation of data including

- an inventory of deployed instances of product versionsWe invite further refinement
- a mapping of vulnerabilities to remediations or mitigations
- a mapping of remediations and/or mitigations to product versions

The first must be collected by the Deployer, while the latter two most often originate from the product Supplier. Managing this information is generally called **asset management**. The relationship between SSVC and asset management is discussed further in Relationship to asset management.

In turn, Deployers must resolve this information into specific actions in which a remediation or mitigation is slated for deployment to replace or modify a particular instance of the product. The Deployer tree in SSVC considers the mission and safety risks inherent to the category of systems to which those deployed instances belong. For this reason, we recommend that the pairing of remediation or mitigation to a product version instance constitutes the unit of work most appropriate for the SSVC.

Coordinator Units of Work

Coordinator units of work tend to coincide with whatever arrives in a single report, which spans the range from a single vulnerability affecting a specific version of an individual product from one Supplier all the way to fundamental design flaws in system specifications that could affect every Supplier and product that uses or implements the flawed specification. Coordinators may need to reorganize reports (e.g., merge, split, expand, or contract) according to their workflow demands. SSVC can be applied to either the initial report or to the results of such refinement.

Aggregation of SSVC across units of work

SSVC users should answer the suggested questions for whatever discrete unit of work they are considering. There is not necessarily a reliable function to aggregate a recommendation about remediation out of its constituent vulnerabilities. For the sake of simplicity of examples, we treat the remediation as a patch of one vulnerability, and comment on any difficulty in generalizing our advice to a more complex patch where appropriate.

To further clarify terms, “Remediation occurs when the vulnerability is eliminated or removed. Mitigation occurs when the impact of the vulnerability is decreased without reducing or eliminating the vulnerability” (Office of the DoD Chief Information Officer 2020, sec. 3.5). Examples of remediation include applying patches, fixes and upgrades; or removing the vulnerable software or system from operation. Mitigating actions may include software configuration changes, adding firewall ACLs, or otherwise limiting the system’s exposure to reduce the risk of the impact of the vulnerability; or accepting the risk.

Supplying Patches

At a basic level, the decision at a software development organization is whether to issue a work order and what resources to expend to remediate a vulnerability in the organization's software. Prioritization is required because, at least in the current history of software engineering, the effort to patch all known vulnerabilities will exceed available resources. The organization considers several other factors to build the patch; refactoring a large portion of the code base may be necessary for some patches, while others require relatively small changes. We focus only on the priority of building the patch, and we consider four categories of priority, as outlined in Table 2.

Table 2: Proposed Meaning for Supplier Priority Outcomes

Supplier Priority	Description
Defer	Do not work on the patch at present.
Scheduled	Develop a fix within regularly scheduled maintenance using supplier resources as normal.
Out-of-Cycle	Develop mitigation or remediation out-of-cycle, taking resources away from other projects and releasing the fix as a security patch when it is ready.
Immediate	Develop and release a fix as quickly as possible, drawing on all available resources, potentially including drawing on or coordinating resources from other parts of the organization.

Deploying Patches

A mitigation that successfully changes the value of a decision point may shift the priority of further action to a reduced state. An effective firewall or IDS rule coupled with an adequate change control process for rules may be enough to reduce the priority where no further action is necessary. In the area of Financial impacts, a better insurance policy may be purchased, providing necessary fraud insurance. Physicial well-being impact may be reduced by testing the physicial barriers designed to restrict a robot's ability to interact with humans. Mission impact could be reduced by correcting the problems identified in a disaster recover test-run of the alternate business flow. If applying a mitigation reduces the priority to *defer*, the deployer may not need to apply a remediation if it later becomes available. Table 3 displays the action priorities for the deployer, which are similar to the supplier case.

When remediation is available, usually the action is to apply it. When remediation is not yet available, the action space is more diverse, but it should involve mitigating the vulnerability (e.g., shutting down services or applying additional security controls) or accepting the risk of not mitigating the vulnerability. Applying mitigations may change the value of decision points. For example, effective firewall and IDS rules may change *System Exposure* from open to controlled. Financial well-being, a *Saftey Impact* category, might be reduced with adequate fraud detection and insurance. Physical well-being, also a *Saftey Impact* category, might be reduced by physical barriers that restrict a robot's ability to interact with humans. *Mission Impact* might be reduced by introducing back-up business flows that do not use the vulnerable component. In a later section we combine Mission and Situated Safety Impact to reduce the complexity of the tree.

However, these mitigation techniques will not always work. For example, the implementation of a firewall or IDS rule to mitigate *System Exposure* from open to controlled is only valid until someone changes the rule. In the area of Financial impacts, the caps on the insurance may be too low to act as a mitigation. The Physical impact may be increased by incorrect installation of the physical barriers designed to restrict a robot's ability to interact with humans. The *Mission Impact* could be increased when a disaster recovery test-run identifies problems with an alternate business flow. The mitigating action may not be permanent or work as designed.

A mitigation that successfully changes the value of a decision point may shift the priority of further action to a reduced state. If applying a mitigation reduces the priority to *defer*, the deployer may not need to apply a remediation, if later, it becomes available. Table 3 displays the action priorities for the deployer, which are similar to the supplier case.

In a later section, the different types of impacts are defined and then implemented in the decision trees as examples of how the various impacts affect the priority. For now, assume the decision points are ordered as: *Exploitation*; *Exposure*; *Utility*; and Well-being and Mission Impact. In this order, an *active* state of *Exploitation* will never result in a *defer* priority. A *none* state of *Exploitation* (no evidence of exploitation) will result in either *defer* or *scheduled* priority—unless the state of Well-being and Mission Impact is *very high*, resulting in an *out-of-cycle* priority.

As opposed to mitigation, applying a remediation finishes an SSVC analysis of a deployed system. While specific vulnerabilities in specific systems can be remediated, the vulnerability cannot be 'disposed of' or eliminated from future consideration within an IT environment. Since software and systems are dynamic, a single vulnerability can be re-introduced after initial remediation through updates, software rollbacks, or other systemic actions that change the operating conditions within an environment. It is therefore important to continually monitor remediated environments for vulnerabilities reintroduced by either rollbacks or new deployments of outdated software.

Table 3: Proposed Meaning for Deployer Priority Outcomes

Deployer Priority	Description
Defer	Do not act at present.
Scheduled	Act during regularly scheduled maintenance time.
Out-of-cycle	Act more quickly than usual to apply the mitigation or remediation out-of-cycle, during the next available opportunity, working overtime if necessary.
Immediate	Act immediately; focus all resources on applying the fix as quickly as possible, including, if necessary, pausing regular organization operations.

Coordinating Patches

In coordinated vulnerability disclosure (CVD), there are two available decisions modelled in SSVC v2. The first is whether or not to coordinate a vulnerability report. This decision is also known as triage. Vulnerability Response Decision Assistance (VRDA) provides a starting point for a decision tree for this situation. VRDA is likely adequate for national-level CSIRTs that do general CVD, but other CSIRT types may have different needs. The CERT guide to CVD provides something similar for those deciding how to report and disclose vulnerabilities they have discovered (Householder, Wassermann, et al. 2020, sec. 6.10). The second is whether to publish information about a vulnerability. We omit a table for this decision because the options are *do not publish* or *publish*.

Table 4: Proposed Coordinator Triage Priority Outcomes

Triage Priority	Description
Decline	Do not act on the report.
Track	Receive information about the vulnerability and monitor for status changes but do not take any overt actions.
Coordinate	Take action on the report. "Action" may include any one or more of: technical analysis, reproduction, notifying vendors, publication, and assist another party.

Items With the Same Priority

Within each setting, the decisions are a kind of equivalence class for priority. That is, if an organization must deploy patches for three vulnerabilities, and if these vulnerabilities are all assigned the *scheduled* priority, then the organization can decide which to deploy first. The priority is equivalent. This approach may feel uncomfortable since CVSS gives the appearance of a finer grained priority. CVSS appears to say, "Not just 4.0 to 6.9 is 'medium' severity, but 4.6 is more severe than 4.5." However, as discussed previously (see page 4), CVSS is designed to be accurate only within +/- 0.5, and, in practice, is scored with errors of around +/- 1.5 to 2.5 (Allodi et al. 2018, see Figure 1). An error of this magnitude is enough to make all of the "normal" range from 4.0 to 6.9 equivalent, because 5.5 +/- 1.5 is the range 4.0 to 7.0. Our proposal is an improvement over this approach. CVSS errors often cross decision boundaries; in other words, the error range often includes the transition between "high" and "critical" or "medium." Since our approach keeps the decisions qualitatively defined, this fuzziness does not affect the results.

Returning to the example of an organization with three vulnerabilities to remediate that were assigned *scheduled* priority, in SSVC, they can be patched in any order. This is an improvement over CVSS, since based on the scoring errors, CVSS was essentially just giving random fine-grained priorities within qualitative categories anyway. With our system, organizations can be more deliberate about conveniently organizing work that is of equivalent priority.

Risk Tolerance and Response Priority

SSVC enables stakeholders to balance and manage their risks themselves. We follow the risk management vocabulary from (ISO 2009) and define risk as "effect of uncertainty on objectives;" see (ISO 2009) for notes on the terms in the definition. A successful vulnerability management practice must balance at least two risks:

1. Change risk: the potential costs of deploying remediation, which include testing and deployment in addition to any problems that could arise from making changes to production systems.
2. Vulnerability risk: the potential costs of incidents resulting from exploitation of vulnerable systems

To place these risks in context, we follow the SEI's Taxonomy of Operational Cyber Security Risks (Cebula and Young 2010). Change risk can be characterized as a combination of Class 2 and/or Class 3 risks. Class 2: Systems and Technology Failures includes hardware, software, and systems risks. Class 3: Failed Internal Processes can arise from process design, process execution, process controls, or supporting processes. Meanwhile, vulnerability risk falls into Subclass 1.2: Actions of People: Deliberate.

In developing the decision trees in this document, we had in mind stakeholders with a moderate tolerance for risk. The resulting trees reflect that assumption. Organizations may of course be more or less conservative in their own vulnerability management practices, and we cannot presume to determine how an organization should balance their risk.

We therefore remind our readers that the labels on the trees (defer, immediate, etc.) can and should be customized to suit the needs of individual stakeholders wherever necessary and appropriate. For example, an organization with a high aversion to change risk might choose to accept more vulnerability risk by lowering the overall response labels for many branches in the trees, resulting in fewer vulnerabilities attaining the most urgent response. On the other hand, an organization with a high aversion to vulnerability risk could elevate the priority of many branches to ensure fixes are deployed quickly.

Scope

Scope is an important variable in the answers of these decision points. It has at least three aspects. The first is how the boundaries of the affected system are set. The second is whose security policy is relevant. The third is how far forward in time or causal steps one reasons about effects and harms. We put forward recommendations for each of these aspects of scope.

However, users of the decision process may want to define different scopes. Users may define a different scope as long as the scope (1) is consistent across decisions, and (2) is credible, explicit, and accessible to all relevant decision makers.

For example, suppliers often decline to support products beyond a declared end-of-life (EOL) date. In these cases, a supplier could reasonably consider vulnerabilities in those products to be out of scope. However, a deployer may still have active instances of EOL products in their infrastructure. It remains appropriate for a deployer to use SSVC to prioritize their response to such situations, since even if there is no remediation forthcoming from the supplier it may be possible for the deployer to mitigate or remediate the vulnerability in other ways, up to and including decommissioning the affected system(s).

Boundaries of the Affected System

One distinction is whether the system of interest is software per se or a cyber-physical system. A problem is that in every practical case, both are involved. Software is what has vulnerabilities and is what vulnerability management is focused on remediating. Multiple pieces of software run on any given computer system. To consider software vulnerabilities in a useful scope, we follow prior work and propose that a vulnerability affects "the set of software instructions that executes in an environment with a coherent function and set of permissions" (Spring, Kern, and Summers 2015). This definition is useful because it lets us keep to common usage and intuition and call the Linux kernel—at least a specific version of it—"one piece" of software.

But decision points about safety and mission impact are not about the software in isolation; they are about the entire cyber-physical system, of which the software is a part. The term "physical" in "cyber-physical system" should be interpreted broadly; selling stocks or manipulating press outlet content are both best understood as affecting human social institutions. These social institutions do not have much of a corporeal instantiation, but they are physical in the sense that they are not merely software, and so are parts of cyber-physical systems.

The hard part of delineating the boundaries of the affected system is specifying what it means for one system to be part of another system. Just because a computer is bolted to a wall does not mean the computer is part of the wall's purpose, which is separating physical space. At the same time, an off-premises DNS server may be part of the site security assurance system if the on-premises security cameras rely on the DNS server to connect to the displays at the guard's desk. We define computer software as part of a cyber-physical system if the two systems are mutually manipulable; that is, changes in the part (the software) will (at least, often) make detectable and relevant changes to the whole (the cyber-physical system), and changes in the whole will (often) make relevant and detectable changes in the part (Spring and Illari 2018).

When reasoning about a vulnerability, we assign the vulnerability to the nearest, relevant—yet more abstract—discrete component. This assignment is particularly important when assessing technical impact on a component. This description bears some clarification, via each of the adjectives:

- **Nearest** is relative to the abstraction at which the vulnerability exists.
- **Relevant** implies that the impacted component must be in the chain of abstraction moving upward from the location of the flaw.
- **More abstract** means that the impacted component is at least one level of abstraction above the specific location of the vulnerability. For example, if the vulnerability is localized to a single line of code in a function, then the function, the module, the library, the application, the product, and the system it belongs to are all "more abstract."
- **Discrete** means there is an identifiable thing that can be remediated (e.g., the unit of patching).

Products, libraries, and applications tend to be appropriate objects of focus when seeking the right level to analyze the impact of a vulnerability. For example, when reasoning about the technical impact of a vulnerability that is localized to a function in a module in an open source library, the nearest relevant discrete abstraction is the library because the patches are made available at the library level. Similarly, analysis of a vulnerability in closed source database software that supports an enterprise resource management (ERM) system would consider the technical impact to the database, not to the ERM system.

Relevant Security Policy

Our definition of a vulnerability includes a security policy violation, but it does not clarify whose security policies are relevant (Householder, Wassermann, et al. 2020). For an organizational PSIRT or CSIRT, the organization's security policy is most relevant. The answer is less clear for coordinators or ISACs. An example scenario that brings the question into focus is phone OS jailbreak methods. The owner of the phone may elect to jailbreak it; there is at least an implicit security policy from the owner that allows this method. However, from the perspective of the explicit phone OS security policy embedded in the access controls and separation of privileges, the jailbreak is exploiting a vulnerability. If a security policy is embedded in technical controls, such as OS access controls on a phone, then anything that violates that security policy is a vulnerability.

Reasoning Steps Forward

This aspect of scope is about immediacy, prevalence, and causal importance. **Immediacy** is about how soon after the decision point adverse effects should occur to be considered relevant. **Prevalence** is about how common adverse effects should be to be considered relevant. **Causal importance** is about how much an exploitation of the software in the cyber-physical system contributes to adverse effects to be considered relevant. Our recommendation is to walk a pragmatic middle path on all three aspects. Effects are not relevant if they are merely possible, too infrequent, far distant, or unchanged by the vulnerability.

But effects are relevant long before they are absolutely certain, ubiquitous, or occurring presently. Overall, we summarize this aspect of scope as *consider credible effects based on known use cases of the software system as a part of cyber-physical systems*.

Likely Decision Points and Relevant Data

We propose the following decision points and associated values should be a factor when making decisions about vulnerability prioritization. Each decision point is tagged with the stakeholder it is relevant to: deployers, suppliers, or both. We emphasize that these descriptions are hypotheses to be further tested and validated. We made every effort to put forward informed and useful decision frameworks with wide applicability, but the goal of this paper is more to solicit feedback than make a declaration. We welcome questions, constructive criticism, refuting evidence, or supporting evidence about any aspect of this proposal.

One important omission from the values for each category is an "unknown" option. Instead, we recommend explicitly identifying an option that is a reasonable assumption based on prior events. Such an option requires reliable historical evidence for what tends to be the case; of course, future events may require changes to these assumptions over time. Therefore, our assumptions require evidence and are open to debate in light of new evidence. Different risk tolerance or risk discounting postures are not addressed in the current work; accommodating such tolerance or discounting explicitly is an area for future work. This flexibility fits into our overall goal of supplying a decision-making framework that is both transparent and fits the needs of different communities. Resisting an "unknown" option discourages the modeler from silently embedding these assumptions in their choices for how the decision tree flows below the selection of any "unknown" option.

We propose satisfactory decision points for vulnerability management in the next sections, in no particular order. Each section has a subsection with advice on gathering information about the decision point. SSVC using Current Information Sources will provide some suggestions about how existing sources of information about vulnerabilities can be used to collate responses to these decision points.

Exploitation

Evidence of Active Exploitation of a Vulnerability

The intent of this measure is the present state of exploitation of the vulnerability. The 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 (Jacobs et al. 2019).

Table 5: Exploitation Decision Values

Value	Definition
None	There is no evidence of active exploitation and no public proof of concept (PoC) of how to exploit the vulnerability.

Value	Definition
PoC (Proof of Concept)	One of the following cases is true: (1) exploit code is sold or traded on underground or restricted fora; (2) a typical public PoC in places such as Metasploit or ExploitDB; or (3) the vulnerability has a well-known method of exploitation. Some examples of condition (3) 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.

Gathering Information About Exploitation

(Householder, Chrabaszcz, et al. 2020) presents a method for searching the GitHub repositories of open-source exploit databases. This method could be employed to gather information about whether PoC is true. However, part (3) of PoC would not be represented in such a search, so more information gathering would be needed. For part (3), perhaps we could construct a mapping of CWE-IDs which always represent vulnerabilities with well-known methods of exploitation. For example, CWE-295, Improper Certificate Validation, and its child CWEs, describe improper validation of TLS certificates. These CWE-IDs could always be marked as PoC since that meets condition (3) in the definition. A comprehensive set of suggested CWE-IDs for this purpose is future work.

Gathering information for active is a bit harder. If the vulnerability has a name or public identifier (such as a CVE-ID), a search of news websites, Twitter, the vendor's vulnerability description, and public vulnerability databases for mentions of exploitation is generally adequate. However, if the organization has the ability to detect exploitation attempts—for instance, through reliable and precise IDS signatures based on a public PoC—then detection of exploitation attempts also signals that active is the right choice. Determining which vulnerability a novel piece of malware uses may be time consuming, requiring reverse engineering and a lot of trial and error. Additionally, capable incident detection and analysis capabilities are required to make reverse engineering possible. Because most organizations do not conduct these processes fully for most incidents, information about which vulnerabilities are being actively exploited generally comes from public reporting by organizations that do conduct these processes. As long as those organizations also share detection methods and signatures, the results are usually quickly corroborated by the community. For these reasons, we assess public reporting by established security community members to be a good information source for active; however, one should not assume it is complete.

The description for none says that there is no **evidence** of active exploitation. This framing admits that an analyst may not be able to detect or know about every attack. An analyst should feel comfortable selecting none if they (or their search scripts) have performed searches in the appropriate places for public PoCs and active exploitation (as described above) and found none. Acknowledging that *Exploitation* values can change relatively quickly, we recommend conducting these searches frequently: if they can be automated to the organization's satisfaction, perhaps once a day (see also Guidance on Communicating Results).

Technical Impact

Technical Impact of Exploiting the Vulnerability

When evaluating *Technical Impact*, recall the scope definition in the Scope Section. Total control is relative to the affected component where the vulnerability resides. If a vulnerability discloses authentication or authorization credentials to the system, this information disclosure should also be scored as "total" if those credentials give an adversary total control of the component.

As mentioned in Current State of Practice, the scope of SSVC is just those situations in which there is a vulnerability. Our definition of **vulnerability** is based on the determination that some security policy is violated. We consider a security policy violation to be a technical impact—or at least, a security policy violation must have some technical instantiation. Therefore, if there is a vulnerability then there must be some technical impact.

Table 6: Technical Impact Decision Values

Value	Definition
Partial	The exploit gives the adversary *limited* control over, or information exposure about, the behavior of the software that contains the vulnerability. Or the exploit gives the adversary an importantly low stochastic opportunity for total control. In this context, "low" means that the attacker cannot reasonably make enough attempts to overcome the low chance of each attempt not working. Denial of service is a form of limited control over the behavior of the vulnerable component.
Total	The exploit gives the adversary *total* control over the behavior of the software, or it gives total disclosure of all information on the system that contains the vulnerability

Gathering Information About Technical Impact

Assessing *Technical Impact* amounts to assessing the degree of control over the vulnerable component the attacker stands to gain by exploiting the vulnerability. One way to approach this analyiss is to ask whether the control gained is *total* or not. If it is not total, it is *partial*. If an answer to one of the following questions is *yes*, then control is *total*. After exploiting the vulnerablily,

- can the attacker install and run arbitrary software?
- can the attacker trigger all the actions that the vulnerable component can perform?
- does the attacker get an account with full privileges to the vulnerable component (administrator or root user accounts, for example)?

This list is an evolving set of heuristics. If you find a vulnerability that should have *total Technical Impact* but that does not answer yes to any of these questions, please describe the example and what question we might add to this list in an issue on the SSVC GitHub.

Utility

The Usefulness of the Exploit to the Adversary

Utility estimates an adversary's benefit compared to their effort based on the assumption that they can exploit the vulnerability. *Utility* is independent from the state of *Exploitation*, which measures whether a set of adversaries have ready access to exploit code or are in fact exploiting the vulnerability. In economic terms, *Exploitation* measures whether the **capital cost** of producing reliable exploit code has been paid or not. *Utility* estimates the **marginal cost** of each exploitation event. More plainly, *Utility* is about how much an adversary might benefit from a campaign using the vulnerability in question, whereas *Exploitation* is about how easy it would be to start such a campaign or if one is already underway.

Heuristically, we base Utility on a combination of the value density of vulnerable components and whether potential exploitation is automatable. This framing makes it easier to analytically derive these categories from a description of the vulnerability and the affected component. *Automatable* as (*no* or *yes*) and *Value Density* as (*diffuse* or *concentrated*) define those decision points.

Roughly, *Utility* is a combination of two things: (1) the value of each exploitation event and (2) the ease and speed with which the adversary can cause exploitation events. We define *Utility* as laborious, efficient, or super effective, as described in Utility Decision Values. The next table is an equivalent expression of *Utility* that resembles a lookup table in a program.

Table 7: Utility Decision Values

Value	Definition
Laborious	*No* to automatable and diffuse value
Efficient	{*Yes* to automatable and diffuse value} OR {*No* to automatable and concentrated value}
Super Effective	*Yes* to automatable and concentrated value

Table 8: Utility to the Adversary, as a Combination of Automatable and Value Density

Automatable	*Value Density*	*Utility*
no	*diffuse*	laborious
no	*concentrated*	efficient
yes	*diffuse*	efficient
yes	*concentrated*	super effective

Automatable

Automatable captures the answer to the question "Can an attacker reliably automate creating exploitation events for this vulnerability?" This metric can take the values *no* or *yes*:

- *no*: Attackers cannot reliably automate steps 1-4 of the kill chain (Hutchins, Cloppert, and Amin 2011) for this vulnerability. These steps are (1) reconnaissance, (2) weaponization, (3) delivery, and (4) exploitation. Reasons why a step may not be reliably automatable could include the following:
  1. the vulnerable component is not searchable or enumerable on the network,
  2. weaponization may require human direction for each target,
  3. delivery may require channels that widely deployed network security configurations block, and
  4. exploitation is not reliable, due to exploit-prevention techniques enabled by default; ASLR is an example of an exploit-prevention tool.
- *yes*: Attackers can reliably automate steps 1-4 of the kill chain. If the vulnerability allows remote code execution or command injection, the expected response should be yes.

Due to vulnerability chaining, there is some nuance as to whether reconnaissance can be automated. For example, consider a vulnerability A. If the systems vulnerable to A are usually not openly connected to incoming traffic (that is, *Exposure* is small or controlled), reconnaissance probably cannot be automated (scans would be blocked, etc.). This would make Automatable equal to no for vulnerability A. However, suppose that another vulnerability B where Automatable is equal to yes can be reliably used to chain to vulnerability A. This automates the *reconnaissance* of vulnerable systems. In this situation, the analyst should continue to analyze vulnerability A to understand whether the remaining steps in the kill chain can be automated.

Gathering Information About Automatable

An analyst should be able to sketch the automation scenario and how it either does or does not satisfy each of the four kill chain steps. Once one step is not satisfied, the analyst can stop and select *no*. Code that demonstrably automates all four kill chain steps certainly satisfies as a sketch. We say sketch to indicate that plausible arguments, such as convincing psuedocode of an automation pathway for each step, are also adequate evidence in favor of a *yes* to *Automatable*.

Like all SSVC decision points, *Automatable* should capture the analyst's best understanding of plausible scenarios at the time of the analysis. An answer of *no* does not mean that it is absolutely inconceivable to automate exploitation in any scenario. It means the analyst is not able to sketch a plausible path through all four kill chain steps. "Plausible" sketches should account for widely deployed network and host-based defenses. Liveness of Internet-connected services means quite a few overlapping things (Bano et al. 2018). For most vulnerabilities, an open port does not automatically mean that reconnaissance, weaponization, and delivery are automatable. Furthermore, discovery of a vulnerable service is not automatable in a situation where only two hosts are misconfigured to expose the service out of 2 million hosts that are properly configured. As discussed in in Reasoning Steps Forward, the analyst should consider *credible* effects based on *known* use cases of the software system to be pragmatic about scope and providing values to decision points.

Value Density

Value Density is described as *diffuse* or *concentrated* based on the resources that the adversary will gain control over with a single exploitation event:

- *diffuse*: The system that contains the vulnerable component has limited resources. That is, the resources that the adversary will gain control over with a single exploitation event are relatively small. Examples of systems with diffuse value are email accounts, most consumer online banking accounts, common cell phones, and most personal computing resources owned and maintained by users. (A "user" is anyone whose professional task is something other than the maintenance of the system or component. As with *Safety Impact*, a "system operator" is anyone who is professionally responsible for the proper operation or maintenance of a system.)
- *concentrated*: The system that contains the vulnerable component is rich in resources. Heuristically, such systems are often the direct responsibility of "system operators" rather than users. Examples of concentrated value are database systems, Kerberos servers, web servers hosting login pages, and cloud service providers. However, usefulness and uniqueness of the resources on the vulnerable system also inform value density. For example, encrypted mobile messaging platforms may have concentrated value, not because each phone's messaging history has a particularly large amount of

data, but because it is uniquely valuable to law enforcement.

Gathering Information About Value Density

The heuristics presented in the *Value Density* definitions involve whether the system is usually maintained by a dedicated professional, although we have noted some exceptions (such as encrypted mobile messaging applications). If there are additional counterexamples to this heuristic, please describe them and the reasoning why the system should have the alternative decision value in an issue on the SSVC GitHub.

An analyst might use market research reports or Internet telemetry data to assess an unfamiliar product. Organizations such as Gartner produce research on the market position and product comparisons for a large variety of systems. These generally identify how a product is deployed, used, and maintained. An organization's own marketing materials are a less reliable indicator of how a product is used, or at least how the organization expects it to be used.

Network telemetry can inform how many instances of a software system are connected to a network. Such telemetry is most reliable for the supplier of the software, especially if software licenses are purchased and checked. Measuring how many instances of a system are in operation is useful, but having more instances does not mean that the software is a densely valuable target. However, market penetration greater than approximately 75% generally means that the product uniquely serves a particular market segment or purpose. This line of reasoning is what supports a determination that an ubiquitous encrypted mobile messaging application should be considered to have a *concentrated* Value Density.

Alternative Utility Outputs

Alternative heuristics can plausibly be used as proxies for adversary utility. One example is the value of the vulnerability if it were sold on the open market. Some firms, such as Zerodium, make such pricing structures public. The valuable exploits track the *Automatable* and *Value Density* heuristics for the most part. Within a single system—whether it is Apache, Windows, iOS or WhatsApp—more successfully automated steps in the kill lead to higher exploit value. Remote code execution with sandbox escape and without user interaction are the most valuable exploits, and these features describe automation of the relevant kill chain steps.

How equivalently *Automatable* exploits for different systems are priced relative to each other is more idiosyncratic. Price does not only track the *Value Density* of the system, but presumably also the existing supply of exploits and the installation distribution among the targets of Zerodium's customers. Currently, we simplify the analysis and ignore these factors. However, future work should look for and prevent large mismatches between the outputs of the *Utility* decision point and the exploit markets.

Safety Impact

Safety Impacts of Affected System Compromise

We take an expansive view of safety, in which a safety violation is a violation of what the United States Centers for Disease Control (CDC) calls **well-being**. Physical well-being violations are common safety violations, but we also consider economic, social, emotional, and psychological well-being to be important. Weighing fine differences among these categories is probably not possible, so we will not try. Each decision option lists examples of the effects that qualify for that value/answer in the various types of violations of well-being. These examples should not be considered comprehensive or exhaustive, but rather as suggestive.

The stakeholder should consider the safety impact on the system operators (by "system operator," we mean those who are professionally responsible for the proper operation of the cyber-physical system, as the term is used in the safety analysis literature) and users of the software they provide. If software is repackaged and resold by a stakeholder to further downstream entities who will then sell a product, the initial stakeholder can only reasonably consider so many links in that supply chain. However, a stakeholder should know its immediate consumers who are one step away in the supply chain. Those consumers may repackage or build on the software and then provide that product further on.

We expect that a stakeholder should be aware of common usage of their software about one step away in the supply chain. This expectation holds in both open source and proprietary contexts. Further steps along the supply chain are probably not reasonable for the stakeholder to consider consistently; however, this is not a license to willfully ignore common downstream uses of the stakeholder's software. If the stakeholder is contractually or legally responsible for safe operation of the software or cyber-physical system of which it is part, that also supersedes our rough supply-chain depth considerations.

For software used in a wide variety of sectors and deployments, the stakeholder may need to estimate an aggregate safety impact. Aggregation suggests that the stakeholder's response to this decision point cannot be less than the most severe credible safety impact, but we leave the specific aggregation method or function as a domain-specific extension for future work.

Gathering Information About Safety Impact

The factors that influence the safety impact level are diverse. This paper does not exhaustively discuss how a stakeholder should answer a question; that is a topic for future work. At a minimum, understanding safety impact should include gathering information about survivability of the vulnerable component, determining available operator actions to compensate for the vulnerable component, understanding relevant insurance, and determining the viability of existing backup measures. Because each of these information items depends heavily on domain-specific knowledge, it is out of the scope of this paper to give a general-purpose strategy for how they should be included. For example, viable manual backup mechanisms are likely to be important in assessing the safety impact of an industrial control system in a sewage plant, but in banking the insurance structures that prevent bankruptcies are more important.

The decision values for safety impact are based on the hazard categories for aircraft software (RTCA, Inc. 2012; FAA 2000, Section 3.3.2). To assign a value to *Safety Impact*, at least one type of harm must reach that value. For example, for a *Safety Impact* of *major*, at least one type of harm must reach *major* level. All types of harm do not need to rise to the level of *major*, just one type of harm does.

Table 9: Safety Impact Decision Values

Value	Type of Harm	Definition
None	All	Does *not* mean no impact *literally*; the effect is below the threshold for all aspects described in Minor
Minor	Physical Harm	Physical discomfort for users of the system OR a minor occupational safety hazard OR reduction in physical system safety margins
Minor	Environment	Minor externalities (property damage, environmental damage, etc.) imposed on other parties

Value	Type of Harm	Definition
Minor	Financial	Financial losses, which are not readily absorbable, to multiple persons
Minor	Psychological	Emotional or psychological harm, sufficient to be cause for counseling or therapy, to multiple persons
Major	Physical Harm	Physical distress and injuries for users of the system OR a significant occupational safety hazard OR failure of physical system functional capabilities that support safe operation
Major	Environment	Major externalities (property damage, environmental damage, etc.) imposed on other parties
Major	Financial	Financial losses that likely lead to bankruptcy of multiple persons
Major	Psychological	Widespread emotional or psychological harm, sufficient to be cause for counseling or therapy, to populations of people
Hazardous	Physical Harm	Serious or fatal injuries, where fatalities are plausibly preventable via emergency services or other measures OR parts of the cyber-physical system that support safe operation break
Hazardous	Environment	Serious externalities (threat to life as well as property, widespread environmental damage, measurable public health risks, etc.) imposed on other parties
Hazardous	Financial	Socio-technical system (elections, financial grid, etc.) of which the affected component is a part is actively destabilized and enters unsafe state
Hazardous	Psychological	N/A
Catastrophic	Physical Harm	Multiple immediate fatalities (emergency response probably cannot save the victims.)
Catastrophic	Environment	Extreme externalities (immediate public health threat, environmental damage leading to small ecosystem collapse, etc.) imposed on other parties
Catastrophic	Financial	Social systems (elections, financial grid, etc.) supported by the software collapse
Catastrophic	Psychological	N/A

Public Safety Impact

Suppliers necessarily have a rather coarse-grained perspective on the broadly defined safety impacts described above. Therefore we simplify the above into a binary categorization: *Significant* is when any impact meets the criteria for an impact of Major, Hazardous, or Catastrophic in the above table. *Minimal* is when none do.

Table 10: Public Safety Impact Decision Values

Value	Definition
Minimal	Safety Impact of None or Minor
Significant	Safety Impact of Major, Hazardous, or Catastrophic

Situated Safety Impact

Deployers are anticipated to have a more fine-grained perspective on the safety impacts broadly defined in Safety Impact. We defer this topic for now because we combine it with *Mission Impact* to simplify implementation for deployers.

Mission Impact

Impact on Mission Essential Functions of the Organization

A **mission essential function (MEF)** is a function "directly related to accomplishing the organization's mission as set forth in its statutory or executive charter" (Fenton 2017, A–1). Identifying MEFs is part of business continuity planning or crisis planning. The rough difference between MEFs and non-essential functions is that an organization "must perform a[n MEF] during a disruption to normal operations" (Fenton 2017, B–2). The mission is the reason an organization exists, and MEFs are how that mission is affected. Non-essential functions do not directly support the mission per se; however, they support the smooth delivery or success of MEFs. Financial losses—especially to publicly traded for-profit corporations—are covered here as a (legally mandated) mission of such corporations is financial performance.

Table 11: Mission Impact Decision Values

Value	Definition
None / Non-Essential Degraded	Little to no impact up to degradation of non-essential functions; chronic degradation would eventually harm essential functions
MEF Support Crippled	Activities that directly support essential functions are crippled; essential functions continue for a time
MEF Failure	Any one mission essential function fails for period of time longer than acceptable; overall mission of the organization degraded but can still be accomplished for a time
Mission Failure	Multiple or all mission essential functions fail; ability to recover those functions degraded; organization's ability to deliver its overall mission fails

Gathering Information About Mission Impact

The factors that influence the mission impact level are diverse. This paper does not exhaustively discuss how a stakeholder should answer a question; that is a topic for future work. At a minimum, understanding mission impact should include gathering information about the critical paths that involve vulnerable components, viability of contingency measures, and resiliency of the systems that support the mission. There are various sources of guidance on how to gather this information; see for example the FEMA guidance in Continuity Directive 2 (Fenton 2017) or OCTAVE FORTE (Tucker 2018). This is part of risk management more broadly. It should require the vulnerability management team to interact with more senior management to understand mission priorities and other aspects of risk mitigation.

As a heuristic, *Utility* might constrain *Mission Impact* if both are not used in the same decision tree. For example, if the *Utility* is *super effective*, then *Mission Impact* is at least *MEF support crippled*.

Human Impact

Combined Situated Safety and Mission Impact

In pilot implementations of SSVC, we received feedback that organizations tend to think of mission and safety impacts as if they were combined into a single factor: in other words, the priority increases regardless which of the two impact factors was increased. We therefore combine Situated Safety and Mission Impact for deployers into a single *Human Impact* factor as a dimension reduction step as follows. We observe that the day-to-day operations of an organization often have already built in a degree of tolerance to small-scale variance in mission impacts. Thus in our opinion we need only concern ourselves with discriminating well at the upper end of the scale. Therefore we combine the three lesser mission impacts of none, non-essential degraded, and MEF support crippled into a single category, while retaining the distinction between MEF Failure and Mission Failure at the extreme. This gives us three levels of mission impact to work with.

On the other hand, most organizations tend to have lower tolerance for variance in safety. Even small deviations in safety are unlikely to go unnoticed or unaddressed. We suspect that the presence of regulatory oversight for safety issues and its absence at the lower end of the mission impact scale influences this behavior. Because of this higher sensitivity to safety concerns, we chose to retain a four-level resolution for the safety dimension. We then combine Mission Impact with Situated Safety impact and map them onto a 4-tiered scale (Low, Medium, High, Very High). The mapping is shown in the following table.

Table 12: Combining Mission and Situated Safety Impact into Human Impact

Situated Safety Impact	Mission Impact	Combined Value (Human Impact)
None/Minor	None/Degraded/Crippled	Low
None/Minor	MEF Failure	Medium
None/Minor	Mission Failure	Very High
Major	None/Degraded/Crippled	Medium
Major	MEF Failure	High
Major	Mission Failure	Very High
Hazardous	None/Degraded/Crippled	High
Hazardous	MEF Failure	High
Hazardous	Mission Failure	Very High
Catastrophic	None/Degraded/Crippled	Very High
Catastrophic	MEF Failure	Very High
Catastrophic	Mission Failure	Very High

Safety and Mission Impact Decision Points for Industry Sectors

We expect to encounter diversity in both safety and mission impacts across different organizations. However, we also anticipate a degree of commonality of impacts to arise across organizations within a given industry sector. For example, different industry sectors may have different use cases for the same software. Therefore, vulnerability information providers—that is, vulnerability databases, Information Sharing and Analysis Organizations (ISAOs), or Information Sharing and Analysis Centers (ISACs)—may provide SSVC information tailored as appropriate to their constituency's safety and mission concerns. For considerations on how organizations might communicate SSVC information to their constituents, see

Guidance on Communicating Results.

System Exposure

The Accessible Attack Surface of the Affected System or Service

Measuring the attack surface precisely is difficult, and we do not propose to perfectly delineate between small and controlled access. Exposure should be judged against the system in its deployed context, which may differ from how it is commonly expected to be deployed. For example, the exposure of a device on a vehicle's CAN bus will vary depending on the presence of a cellular telemetry device on the same bus.

If a vulnerability cannot be remediated, other mitigations may be used. Usually, the effect of these mitigations is to reduce exposure of the vulnerable component. Therefore, a deployer's response to Exposure may change if such mitigations are put in place. If a mitigation changes exposure and thereby reduces the priority of a vulnerability, that mitigation can be considered a success. Whether that mitigation allows the deployer to defer further action varies according to each case.

Table 13: System Exposure Decision Values

Value	Definition
Small	Local service or program; highly controlled network
Controlled	Networked service with some access restrictions or mitigations already in place (whether locally or on the network). A successful mitigation must reliably interrupt the adversary's attack, which requires the attack is detectable both reliably and quickly enough to respond. *Controlled* covers the situation in which a vulnerability can be exploited through chaining it with other vulnerabilities. The assumption is that the number of steps in the attack path is relatively low; if the path is long enough that it is implausible for an adversary to reliably execute it, then *exposure* should be *small*.
Open	Internet or another widely accessible network where access cannot plausibly be restricted or controlled (e.g., DNS servers, web servers, VOIP servers, email servers)

Gathering Information About System Exposure

System Exposure is primarily used by Deployers, so the question is about whether some specific system is in fact exposed, not a hypothetical or aggregate question about systems of that type. Therefore, it generally has a concrete answer, even though it may vary from vulnerable component to vulnerable component, based on their respective configurations.

System Exposure can be readily informed by network scanning techniques. For example, if the vulnerable component is visible on Shodan or by some other external scanning service, then it is *open*. Network policy or diagrams are also useful information sources, especially for services intentionally open to the Internet such as public web servers. An analyst should also choose *open* for a phone or PC that connects to the web or email without the usual protections (IP and URL blocking, updated firewalls, etc.).

Distinguishing between *small* and *controlled* is more nuanced. If *open* has been ruled out, some suggested heuristics for differentiating the other two are as follows. Apply these heuristics in order and stop when one of them applies.

- If the system's networking and communication interfaces have been physically removed or disabled, choose *small*.
- If *Automatable* is *yes*, then choose *controlled*. The reasoning behind this heuristic is that if reconnaissance through exploitation is automatable, then the usual deployment scenario exposes the system sufficiently that access can be automated, which contradicts the expectations of *small*.
- If the vulnerable component is on a network where other hosts can browse the web or receive email, choose *controlled*.

If you have suggestions for further heuristics, or potential counterexamples to these, please describe the example and reasoning in an issue on the SSVC GitHub.

Decisions During Vulnerability Coordination

Coordinators are facilitators within the vulnerability management ecosystem. Since coordinators neither supply nor deploy the vulnerable component in question, their decisions are different from suppliers' or deployers' decisions. This section provides a window into CERT/CC's decisions as an example of how a coordinator might use SSVC to make its own decisions.

Coordinators vary quite a lot, and their use of SSVC may likewise vary. A coordinator may want to gather and publish information about SSVC decision points that it does not use internally in order to assist its constituents. Furthermore, a coordinator may only publish some of the information it uses to make decisions. Consistent with other stakeholder perspectives (supplier and deployer), SSVC provides the priority with which a coordinator should take some defined action, but not how to do that action. For more information about types of coordinators and their facilitation actions within vulnerability management, see (Householder, Wassermann, et al. 2020).

The two decisions that CERT/CC makes as a coordinator that we will discuss in terms of SSVC are the initial triage of vulnerability reports and whether a publication about a vulnerability is warranted. The initial coordination decision is a prioritization decision, but it does not have the same values as prioritization by a deployer or supplier. The publication decision for us is a binary yes/no. These two decisions are not the entirety of vulnerability coordination, but we leave further details of the process for future work.

Different coordinators have different scopes and constituencies. See (Householder, Wassermann, et al. 2020, 3.5) for a listing of different coordinator types. If a coordinator receives a report that is outside its own work scope or constituency, it should make an effort to route the report to a more suitable coordinator. The decisions in this section assume the report or vulnerability in question is in the work scope or constituency for the coordinator.

Coordination Triage Decisions

We take three priority levels in our decision about whether and how to coordinate a vulnerability (Householder, Wassermann, et al. 2020, 1.1) based on an incoming report:

- *Decline*—Do not act on the report. May take different forms, including ignoring the report as well as an acknowledgement to the reporter that we will not act and suggest the reporter to go to vendor or publish if unresponsive.

- *Track*—Receive information about the vulnerability and monitor for status changes but do not take any overt actions.
- *Coordinate*—Take action on the report. "Action" may include any one or more of: technical analysis, reproduction, notifying vendors, lead coordination (notify, communicate, and publish), publish only (amplify public message), advise only, secondary coordinator (assist another lead coordinator). See (Benetis et al. 2019) for additional vulnerability management services a coordinator may provide.

Coordinator Decision Points

Our goal with the coordination decision is to base it on information that is available to the analyst when CERT/CC receives a vulnerability report. In addition to using some of the decision points in Likely Decision Points; coordination makes use of Utility and Public Safety Impact decision points. The coordination and publication decisions for CERT/CC are about the social and collaborative state of vulnerability management. To assess this, the decision involves five new decision points.

Report Public

Is a viable report of the details of the vulnerability already publicly available?

- Yes
- No

Supplier Contacted

Has the reporter made a good-faith effort to contact the supplier of the vulnerable component using a quality contact method?

- Yes
- No

A quality contact method is a publicly posted known good email address, public portal on vendor website, etc.

Supplier Cardinality

How many suppliers are responsible for the vulnerable component and its remediation or mitigation plan?

- One
- Multiple

Supplier Engagement

Is the supplier responding to the reporter's contact effort and actively participating in the coordination effort?

- Active
- Unresponsive

Report Credibility

Assessing the credibility of a report is complex, but the assessment must reach a conclusion of either:

- Credible
- Not credible

An analyst should start with a presumption of credibility and proceed toward disqualification. The reason for this is that, as a coordinator, occasionally doing a bit of extra work on a bad report is preferable to rejecting legitimate reports. This is essentially stating a preference for false positives over false negatives with respect to credibility determination.

The following subsections provide suggestive guidance on assessing credibility. There are no ironclad rules for this assessment, and other coordinators may find other guidance works for them. Credibility assessment topics include indicators for and against credibility, perspective, topic, and relationship to report scope.

Credibility Indicators The credibility of a report is assessed by a balancing test. The indicators for or against are not commensurate, and so they cannot be put on a scoring scale, summed, and weighed.

A report may be treated as credible when either (1) the vendor confirms the existence of the vulnerability or (2) independent confirmation of the vulnerability by an analyst who is neither the reporter nor the vendor. If neither of these confirmations are available, then the value of the *Report Credibility* decision point depends on a balancing test among the following indicators.

Indicators *for* Credibility include:

- The report is specific about what is affected
- The report provides sufficient detail to reproduce the vulnerability.
- The report describes an attack scenario.
- The report suggests mitigations.
- The report includes proof-of-concept exploit code or steps to reproduce.
- Screenshots and videos, if provided, support the written text of the report and do not replace it.
- The report neither exaggerates nor understates the impact.

Indicators *against* Credibility include:

- The report is “spammy” or exploitative (for example, the report is an attempt to upsell the receiver on some product or service).
- The report is vague or ambiguous about which vendors, products, or versions are affected (for example, the report claims that all “cell phones” or “wifi” or “routers” are affected).
- The report is vague or ambiguous about the preconditions necessary to exploit the vulnerability.
- The report is vague or ambiguous about the impact if exploited.
- The report exaggerates the impact if exploited.
- The report makes extraordinary claims without correspondingly extraordinary evidence (for example, the report claims that exploitation could result in catastrophic damage to some critical system without a clear causal connection between the facts presented and the impacts claimed).
- The report is unclear about what the attacker gains by exploiting the vulnerability. What do they get that they didn’t already have? For example, an attacker with system privileges can already do lots of bad things, so a report that assumes system privileges as a precondition to exploitation needs to explain what else this gives the attacker.
- The report depends on preconditions that are extremely rare in practice, and lacks adequate evidence for why those preconditions might be expected to occur (for example, the vulnerability is only exposed in certain non-default configurations—unless there is evidence that a community of practice has established a norm of such a non-default setup).

- The report claims dire impact for a trivially found vulnerability. It is not impossible for this to occur, but most products and services that have been around for a while have already had their low-hanging fruit major vulnerabilities picked. One notable exception would be if the reporter applied a completely new method for finding vulnerabilities to discover the subject of the report.
- The report is rambling and is more about a narrative than describing the vulnerability. One description is that the report reads like a food recipe with the obligatory search engine optimization preamble.
- The reporter is known to have submitted low-quality reports in the past.
- The report conspicuously misuses technical terminology. This is evidence that the reporter may not understand what they are talking about.
- The analyst's professional colleagues consider the report to be not credible.
- The report consists of mostly raw tool output. Fuzz testing outputs are not vulnerability reports.
- The report lacks sufficient detail for someone to reproduce the vulnerability.
- The report is just a link to a video or set of images, or lacks written detail while claiming "it's all in the video". Imagery should support a written description, not replace it.
- The report describes a bug with no discernible security impact.
- The report fails to describe an attack scenario, and none is obvious.

We considered adding poor grammar or spelling as an indicator of non-credibility. On further reflection, we do not recommend that poor grammar or spelling be used as an indicator of low report quality, as many reporters may not be native to the coordinator's language. Poor grammar alone is not sufficient to dismiss a report as not credible. Even when poor grammar is accompanied by other indicators of non-credibility, those other indicators are sufficient to make the determination.

Credibility of what to whom SSVC is interested in the coordinating analyst's assessment of the credibility of a report. This is separate from the fact that a reporter probably reports something because they believe the report is credible.

The analyst should assess the credibility of the report of the vulnerability, not the claims of the impact of the vulnerability. A report may be credible in terms of the fact of the vulnerability's existence even if the stated impacts are inaccurate. However, the more extreme the stated impacts are, the more skepticism is necessary. For this reason, "exaggerating the impact if exploited" is an indicator *against* credibility. Furthermore, a report may be factual but not identify any security implications; such reports are bug reports, not vulnerability reports, and are considered out of scope.

A coordinator also has a scope defined by their specific constituency and mission. A report can be entirely credible yet remain out of scope for your coordination practice. Decide what to do about out of scope reports separately, before the vulnerability coordination triage decision begins. If a report arrives and would be out of scope even if true, there will be no need to proceed with judging its credibility.

Coordination Triage Decision Process

The decision tree for reaching a Decision involves seven decision points. The first two function as gating questions:

- If a report is already public, then CERT/CC will decline the case unless there are multiple suppliers, *super effective Utility*, and *significant* Public Safety Impact.

- If no suppliers have been contacted, then CERT/CC will decline the case unless there are multiple suppliers, *super effective Utility*, and *significant* Public Safety Impact.

In the second case, CERT/CC may encourage the reporter to contact the supplier and submit a new case request if the supplier is unresponsive.

These two sets of exceptional circumstances mean that the seven decision points involved in the coordination triage tree can be compressed slightly, as the tree shows. This tree's information is available as either a CSV or PDF

Publication Decision

A coordinator often has to decide when or whether to publish information about a vulnerability. A supplier also makes a decision about publicity—releasing information about a remediation or mitigation could be viewed as a kind of publication decision. While the context of publication is different for coordinators, a supplier may find some use in a publication decision if they need to decide whether to publish before a mitigation or remediation is available. However, that is not the intended use case; this section describes how a coordinator might decide to publish information about a vulnerability.

The publication decision is always a decision at a point in time. As discussed in Guidance on Communicating Results, a SSVC decision may change over time as the inputs to that decision change. A decision to publish cannot be revoked, since the publication is likely to be archived or at least remembered. However, a decision to not publish is a decision not to publish at the time the decision was made. It is not a decision never to publish in the future. One benefit of encoding the decision process in SSVC is the analysts can all agree on what new information would change the decision and prioritize maintaining awarenss of just those decision points.

This section is based on CERT/CC policy choices. Two points where this clearly influences the publication decision are embargo periods and scope. As a matter of policy, CERT/CC will support an embargo from the public of information about a vulnerability through its choice not to publish that information while a number of conditions hold:

- A negotiated embargo timer has not expired. The CERT/CC default embargo period is 45 days.
- Other exceptions have not been met, including active exploitation of the vulnerability in the wild or other public discussion of the vulnerability details. Regardless, the decision described in this section assumes the embargo period is over, one way or another. The second point is related to the Coordination Triage Decisions and the status of the vulnerability. CERT/CC only expects to publish about vulnerabilities with a *coordinate* status. While an issue that is tracked or declined may be reevaluated at a later date and status changed to *coordinate*, unless that happens we would not publish about the vulnerability. Other organizations, such as NVD, would have different publication criteria and may want to include decision points or the decision itself from the Coordination Triage Decisions in their publication decision.

In addition to the two decision points defined in this section, the publication decision uses the *Exploitation* decision point.

Public Value Added

This decision point asks how much value a publication from the coordinator would benefit the broader community. The value is based on the state of existing public information about the vulnerablity.

- *Precedence*—the publication would be the first publicly available, or be coincident with the first publicly available.
- *Ampliative*—amplifies and/or augments the existing public information about the vulnerability, for example, adds additional detail, addresses or corrects errors in other public information, draws further attention to the vulnerability, etc.
- *Limited*—minimal value added to the existing public information because existing information is already high quality and in multiple outlets.

The intent of the definition is that one rarely if ever transitions from limited to ampliative or ampliative to precedence. A vulnerability could transition from precedence to ampliative and ampliative to limited. That is, *Public Value Added* should only be downgraded through future iterations or re-evaluations. This directionality is because once other organizations make something public, they cannot effectively un-publish it (it'll be recorded and people will know about it, even if they take down a webpage). The rare case where *Public Value Added* increases would be if an organization published viable information, but then published additional misleading or obscuring information at a later time. Then one might go from *limited* to *ampliative* in the interest of pointing to the better information.

Supplier Involvement

This decision point accounts for the state of the supplier's work on addressing the vulnerability.

- *Fix Ready*—the supplier has provided a patch or fix
- *Cooperative*—the supplier is actively generating a patch or fix; they may or may not have provided a mitigation or work-around in the mean time.
- *Uncooperative/Unresponsive*—the supplier has not responded, declined to generate a remediation, or no longer exists.

Prioritization

Given a specific stakeholder decision and set of useful decision points, we are now in a position to combine them into a comprehensive set of decisions about the priority with which to act. The definition of choices can take a logical form, such as:

- IF
- (*Exploitation* IS PoC) AND
- (*Exposure* IS controlled) AND
- (*Utility* IS laborious) AND
- (*Well-being and Mission Impact* IS medium)
- THEN priority is *scheduled*.

This logical statement is captured in line 50 of the deployer .csv file.

There are different formats for capturing these prioritization decisions depending on how and where they are going to be used. In this paper, we primarily represent a full set of guidance on how one stakeholder will make a decision as a **decision tree**. This section presents example trees for each stakeholder: supplier, deployer, and coordinator. This section also provides some guidance on how to construct and customize a decision tree and gather evidence to make decisions. How this decision information might be stored or communicated is the topic of subsections on Asset Management and Communication.

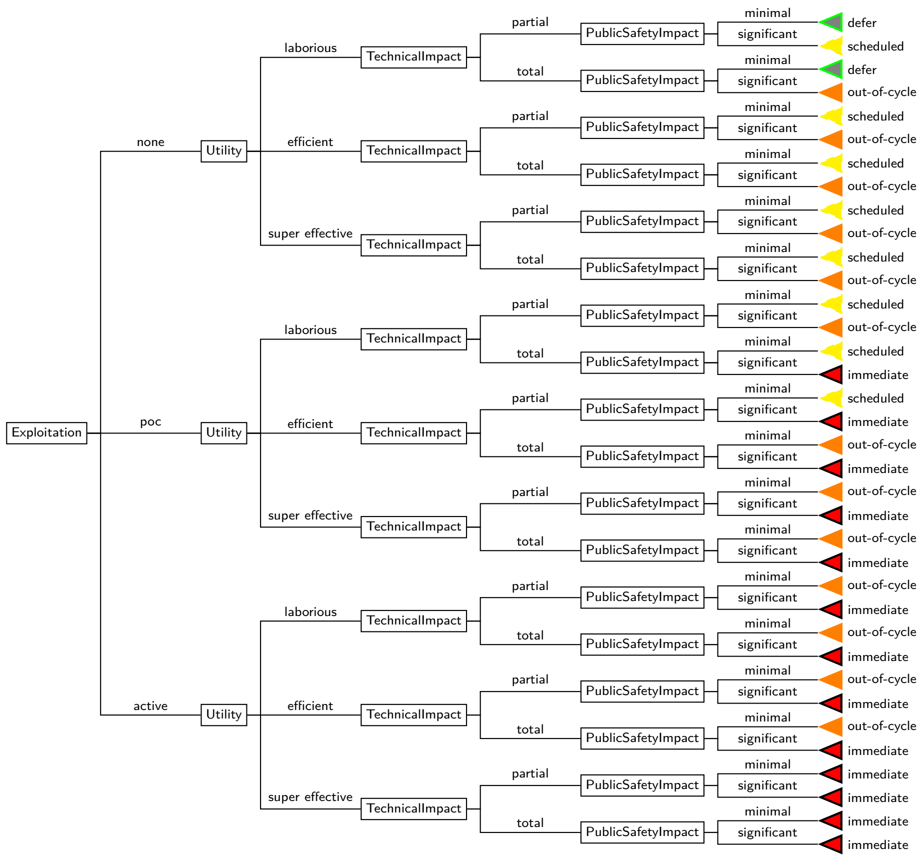

Figure 1: Suggested Supplier Tree

Supplier Tree

The example supplier tree PDF shows the proposed prioritization decision tree for the supplier. Both supplier and deployer trees use the above decision point definitions. Each tree is a compact way of expressing assertions or hypotheses about the relative priority of different situations. Each tree organizes how we propose a stakeholder should treat these situations. Rectangles are decision points, and triangles represent outcomes. The values for each decision point are different, as described above. Outcomes are priority decisions (defer, scheduled, out-of-cycle, immediate); outcome triangles are color coded:

- Defer = gray with green outline
- Scheduled = yellow
- Out-of-Cycle = orange
- Immediate = red with black outline

Deployer Tree

The example deployer tree PDF is depicted below.

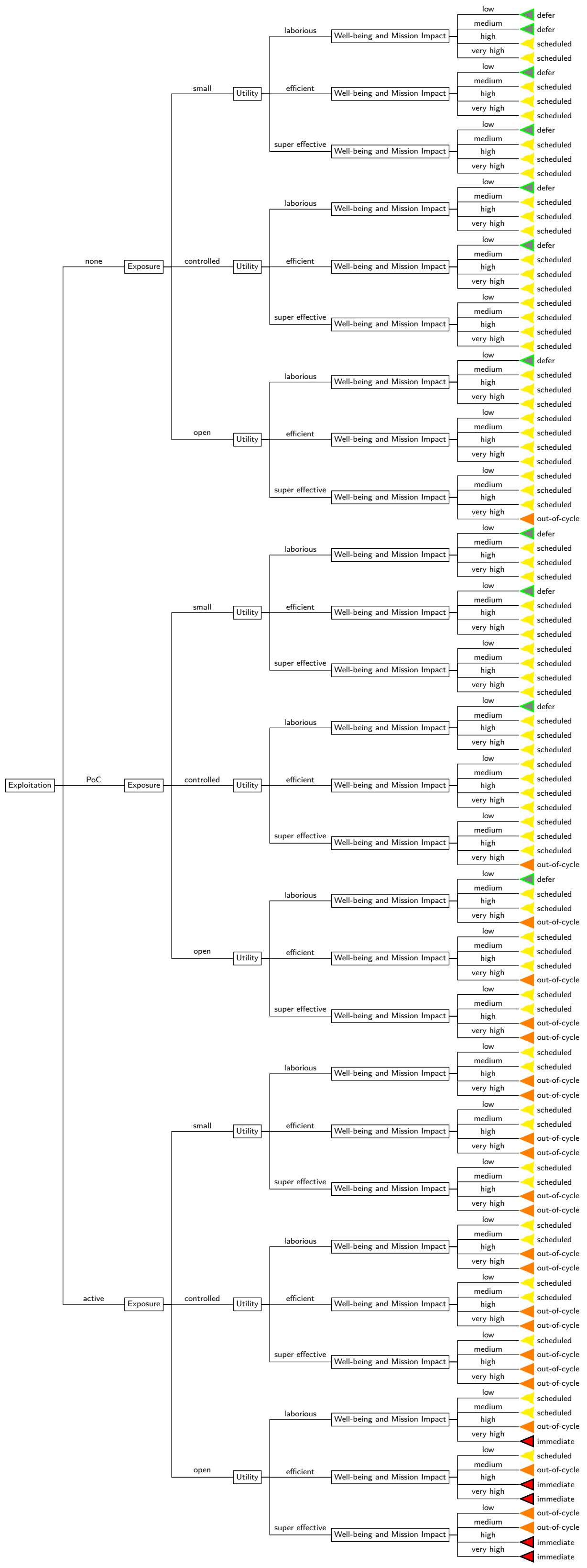

Figure 2: Suggested Deployer Tree

Coordinator trees

As described in Decisions During Vulnerability Coordination, a coordination stakeholder usually makes separate triage and publication decisions. Each have trees presented below.

Triage decision tree

This tree is a suggestion in that CERT/CC believes it works for us. Other coordinators should consider customizing the tree to their needs, as described in Tree Construction and Customization Guidance.

Publication decision tree

Suggested decision values for this decision are available in CSV and PDF formats.

Tree Construction and Customization Guidance

Stakeholders are encouraged to customize the SSVC decision process to their needs. Indeed, the first part of SSVC stands for "stakeholder-specific." However, certain parts of SSVC are more amenable to customization than others. In this section, we'll cover what a stakeholder should leave static, what we imagine customization looks like, and some advice on building a usable and manageable decision tree based on our experience so far.

We suggest that the decision points, their definitions, and the decision values should not be customized. Different vulnerability management teams inevitably think of topics such as *Utility* to the adversary in slightly different ways. However, a key contribution of SSVC is enabling different teams to communicate about their decision process. In order to clearly communicate differences in the process, the decision points that factor into the process need to be the same between different teams. A stakeholder community may come together and, if there is broad consensus, add or change decision points.

Which decision points are involved in a vulnerability management team's decision and the priority label for each resulting situation are, for all intents and purposes, totally at the discretion of the team. We have provided some examples for different stakeholder communities here. What decision points a team considers reflects what it cares about and the risks prioritizes. Different teams may legitimately prioritize different objectives. It should be easier for teams to discuss and communicate such differences if the definitions of the decision points remain static. The other aspect of risk management that SSVC allows a team to customize is its risk appetite or risk tolerance.

A team's risk appetite is reflected directly by the priority labels for each combination of decision values. For example, a vulnerability with no or minor *Public Safety Impact*, total *Technical Impact*, and efficient *Utility* might be handled with *scheduled* priority by one team and *out-of-cycle* priority by another. As long as each team has documented this choice and is consistent in its own application of its own choice, the two teams can legitimately have different appetites for vulnerability risk. SSVC enables teams with such different risk appetites to discuss and communicate precisely the circumstances where they differ.

When doing the detailed risk management work of creating or modifying a tree, we recommend working from text files with one line or row for each unique combination of decision values. For examples, see SSVC/data. An important benefit, in our experience, is that it's easier to identify a question by saying "I'm unsure about row 16" than anything else we have thought of so far. Once the humans agree on the decision tree, it can be converted to a JSON schema for easier machine-readable communication, following the provided SSVC provision JSON schema.

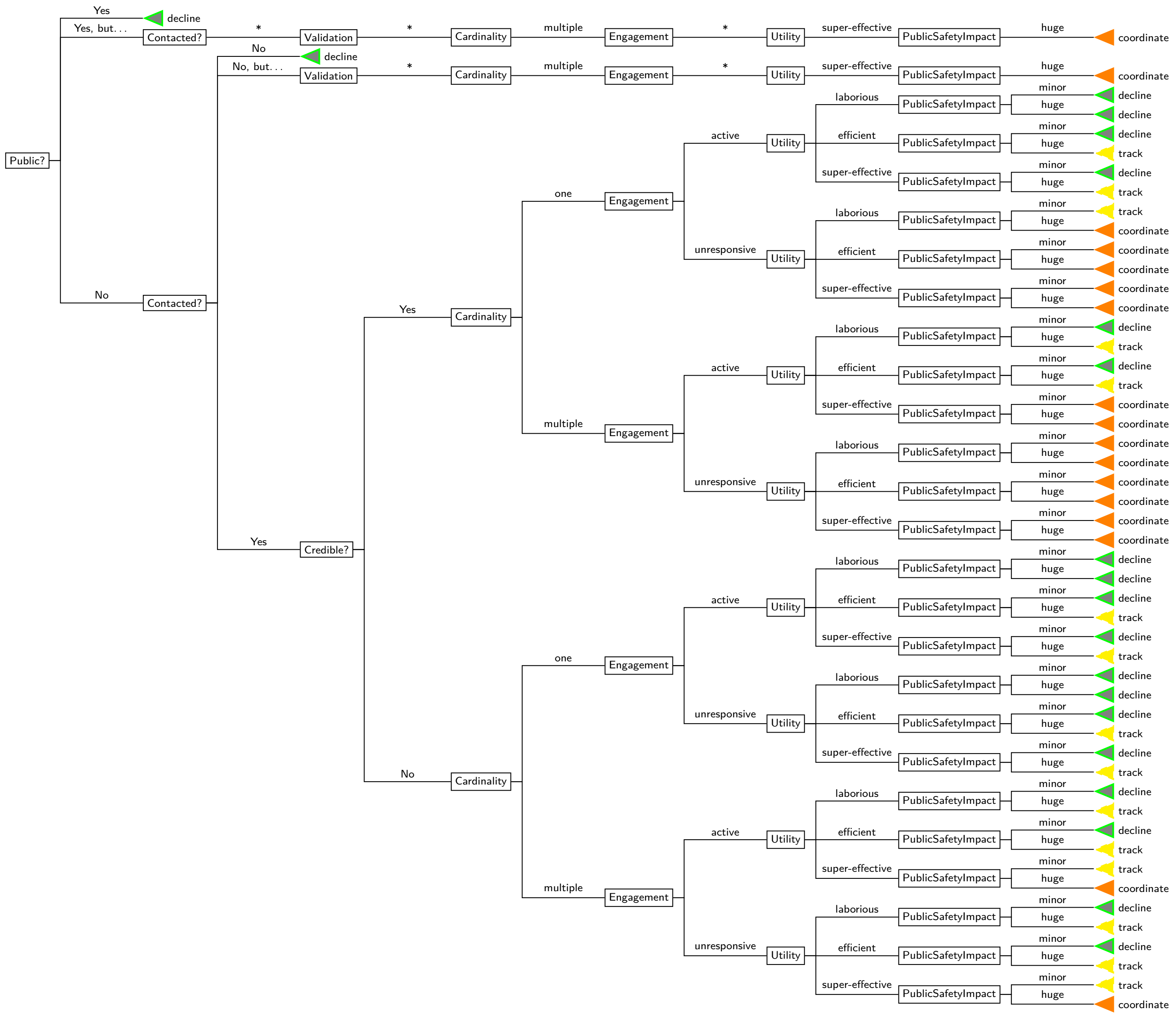

Figure 3: Suggested Coordinator Triage Tree

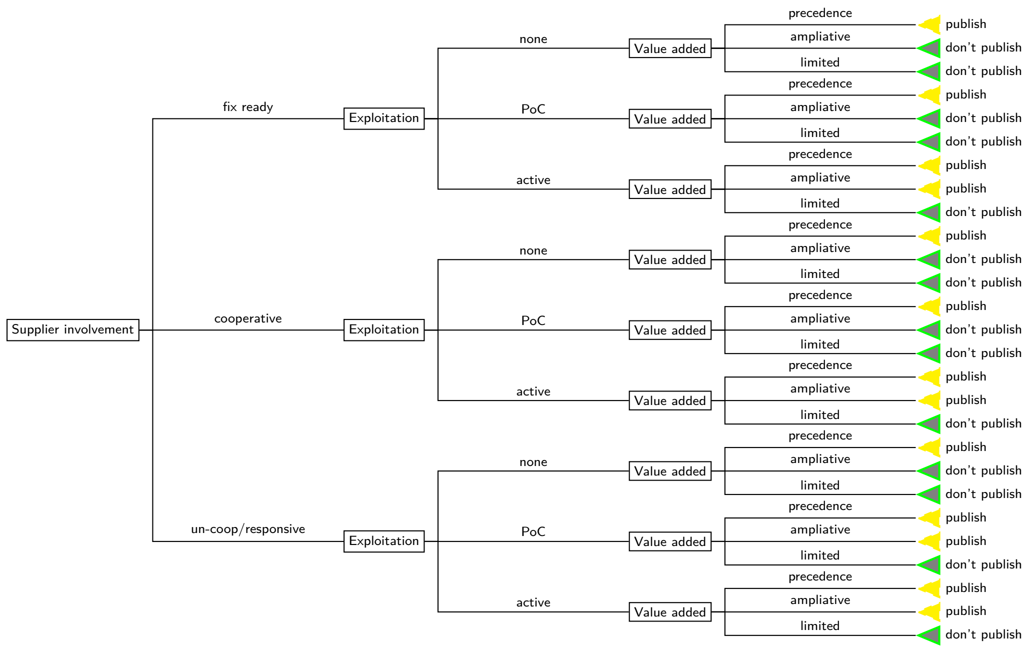

Figure 4: Suggested Coordinator Publication Tree

Once the decision points are selected and the prioritization labels agreed upon, it is convenient to be able to visually compress the text file by displaying it as a decision tree. Making the decision process accessible has a lot of benefits. Unfortunately, it also makes it a bit too easy to overcomplicate the decision.

The academic literature surrounding the measurement of decision tree quality is primarily concerned with measuring classification errors given a particular tree and a labeled data set. In our case, we are not attempting to fit a tree to data. Rather, we are interested in producing usable trees that minimize extraneous effort. To that end, we briefly examine the qualities for which decision tree measurement is suitable.

Decision tree construction methods must address four significant concerns: feature selection, feature type, overfitting, and parsimony.

Feature selection is perhaps the most important consideration for SSVC, because it directly affects the information gathering requirements placed on the analyst attempting to use the tree. Each decision point in SSVC is a feature.

The SSVC version 1 ~applier~ deployer tree had 225 rows when we wrote it out in long text form. It only has four outcomes to differentiate between. Thus, on average that decision process treats one situation (combination of decision values) as equivalent to 65 other situations. If nothing else, this means analysts are spending time gathering evidence to make fine distinctions that are not used in the final decision. The added details also make it harder for the decision process to accurately manage the risks in question. This difficulty arises because more variance and complexity there is in the decision increases the possibility of errors in the decision process itself.

Regarding feature types, all of the features included in SSVC v2 can be considered ordinal data. That is, while they can be ordered (e.g., for Exploitation, active is greater than poc is greater than none), they can not be compared via subtraction or division (active - poc = nonsense). The use of ordinal features is a key assumption behind our use of the parsimony analysis that follows.

When decision trees are used in a machine learning context, overfitting increases tree complexity by incorporating the noise in the training data set into the decision points in a tree. In our case, our "data" is just the set of outcomes as decided by humans, so overfitting is less of a concern, assuming the feature selection has been done with care.

Parsimony is, in essence, Occam's Razor applied to tree selection. Given the choice between two trees that have identical outputs, one should choose the tree with fewer decisions. One way to evaluate the parsimony of a tree is by applying the concept of feature importance to ensure that each feature is contributing adequately to the result. While there are a few ways to compute feature importance, the one we found most useful is permutation importance. Permutation importance can be calculated on a candidate tree to highlight potential issues. It works by randomly shuffling the values for each feature individually and comparing a fitness metric on the shuffled tree to the original. The change in fitness is taken to be the importance of the feature that was shuffled. Permutation importance is usually given as a number in the interval [0,1]. Python's scikit-learn provides a permutation importance method, which we used to evaluate our trees.

Interpreting the results of a permutation importance computation on a tree involves nuance, but one rule we can state is this: any feature with a computed permutation importance of zero can be eliminated from the tree without losing any relevant information. When all of the permutation importance scores for all features are relatively equal, that is an indication that each feature is approximately equally relevant to the decision.

More likely, however, is that some subset of features will be of relatively equal importance, and one might be of considerably lower importance (yet not zero). In this case, the lowest importance feature should be considered for refinement or elimination. It is possible that adjusting the definition of a feature or its available values (whether redefining, adding, or removing options) could increase its importance. Reasons to retain a low-importance feature include:

- the feature is relevant to a small set of important circumstances that a tree without the feature would otherwise be unable to discriminate
- the effort required to determine the correct value for the feature is relatively small, for example information that might be collected automatically
- the feature enables other features to be defined more clearly Features that meet none of the above criteria may be good candidates for elimination.

Customizing a tree by changing the outcome priority labels can also affect the importance of a feature. This sort of customization is often the simplest way to adjust the importance of a feature.

While there is no hard and fast rule for when a tree is too big, we suggest that if all of your outcomes are associated with more than 15 situations (unique combinations of decision values), you would benefit from asking whether your analysts actually use all the information they would be gathering. Thus, 60 unique combinations of decision values is the point at which a decision tree with four distinct outcomes is, on average, potentially too big.

SSVC trees should be identifiable by name and version. A tree name is simply a short descriptive label for the tree derived from the stakeholder and/or function the tree is intended for. Tree versions are expected to share the major and minor version numbers with the SSVC version in which their decision points are defined. Revisions should increment the patch number. For example: "Applier Tree v1.1.0" would be the identity of the version of the Applier Tree as published in v1.1 of SSVC. “Coordinator Publish Tree v2.0.3” would be the identity of a future revision of the Coordinator Publish Tree as described in this document. The terms “major”, “minor”, and “patch” with respect to version numbering are intended to be consistent with Semantic Versioning 2.0.0.

Guidance for Evidence Gathering

To answer each of these decision points, a supplier or deployer should, as much as possible, have a repeatable evidence collection and evaluation process. However, we are proposing decisions for humans to make, so evidence collection and evaluation is not totally automatable. That caveat notwithstanding, some automation is possible.

For example, whether exploitation modules are available in ExploitDB, Metasploit, or other sources is straightforward. We hypothesize that searching Github and Pastebin for exploit code can be captured in a script. A supplier or deployer could then define *Exploitation* to take the value of *PoC* if there are positive search results for a set of inputs derived from the CVE entry in at least one of these venues. At least, for those vulnerabilities that are not “automatically” PoC-ready, such as on-path attackers for TLS or network replays.

Some of the decision points require a substantial upfront analysis effort to gather risk assessment or organizational data. However, once gathered, this information can be efficiently reused across many vulnerabilities and only refreshed occasionally. An obvious example of this is the mission impact decision point. To answer this, a deployer must analyze their essential functions, how they interrelate, and how they are supported. Exposure is similar; answering that decision point requires an asset inventory, adequate understanding of the network topology, and a view of the enforced security controls. Independently operated scans, such as Shodan or Shadowserver, may play a role in evaluating exposure, but the entire exposure question cannot be reduced to a binary question of whether an organization’s assets appear in such databases. Once the deployer has the situational awareness to understand MEFs or exposure, selecting the answer for each individual vulnerability is usually straightforward.

Stakeholders who use the prioritization method should consider releasing the priority with which they handled the vulnerability. This disclosure has various benefits. For example, if the supplier publishes a priority ranking, then deployers could consider that in their decision-making process. One reasonable way to include it is to break ties for the deployer. If a deployer has three “scheduled” vulnerabilities to remediate, they may address them in any order. If two vulnerabilities were produced by the supplier as “scheduled” patches, and one was “out-of-cycle,” then the deployer may want to use that information to favor the latter.

In the case where no information is available or the organization has not yet matured its initial situational analysis, we can suggest something like defaults for some decision points. If the deployer does not know their exposure, that means they do not know where the devices are or how they are controlled, so they should assume *System Exposure* is *open*. If the decision maker knows nothing about the environment in which the device is used, we suggest assuming a *major Safety Impact*. This position is conservative, but software is thoroughly embedded in daily life now, so we suggest that the decision maker provide evidence that no one’s well-being will suffer. The reach of software exploits is no longer limited to a research network. Similarly, with *Mission Impact*, the deployer should assume that the software is in use at the organization for a reason, and that it supports essential functions unless they have evidence otherwise. With a total lack of information, assume *support crippled* as a default. *Exploitation* needs no special default; if adequate searches are made for exploit code and none is found, the answer is *none*. If nothing is known about *Automatable*, the safer answer to assume is *yes*. *Value Density* should always be answerable; if the product is uncommon then it is probably *diffuse*. The resulting decision set {*none*, *open*, *efficient*, *medium*} results in a scheduled patch application in our recommended deployer tree.

Relationship to asset management

Vulnerability management is a part of asset management. SSVC can benefit from asset management practices and systems, particularly in regard to automating data collection and answers for some decision points. SSVC depends on asset management to some extent, particularly for context on the cost and risk associated with changing or updating the asset.

Asset management can help automate the collection of the *Mission Impact*, *Situated Safety Impact*, and *System Exposure* decision points. These decision points tend to apply per asset rather than per vulnerability. Therefore, once each is assessed for each asset, it can be applied to each vulnerability that applies to that asset. While the asset assessment should be reviewed occasionally for accuracy, storing this data in an asset management system should enable automated scoring of new vulnerabilities on these decision points for those assets.

Our method is for prioritizing vulnerabilities based on the risk stemming from exploitation. There are other reasonable asset management considerations that may influence remediation timelines. There are at least three aspects of asset management that may be important but are out of scope for SSVC. First and most obvious is the transaction cost of conducting the mitigation or remediation. System administrators are paid to develop or apply any remediations or mitigations, and there may be other transactional costs such as downtime for updates. Second is the risk of the remediation or mitigation introducing a new error or vulnerability. Regression testing is part of managing this type of risk. Finally, there may be an operational cost of applying a remediation or mitigation, representing an ongoing change of functionality or increased overhead. A decision maker could order work within one SSVC priority class (scheduled, out-of-cycle, etc.) based on these asset management considerations, for example. Once the organization remediates or mitigates all the high-priority vulnerabilities, they can then focus on the medium-level vulnerabilities with the same effort spent on the high-priority ones.

Asset management and risk management also drive some of the up-front work an organization would need to do to gather some of the necessary information. This situation is not new; an asset owner cannot prioritize which fixes to deploy to its assets if it does not have an accurate inventory of its assets. The organization can pick its choice of tools; there are about 200 asset management tools on the market (Captera 2019). Emerging standards like the Software Bill of Materials (SBOM) (Jump and Manion 2019) would likely reduce the burden on asset management, and organizations should prefer systems which make such information available. If an organization does not have an asset management or risk management (see also Gathering Information About Mission Impact) plan and process in place, then SSVC provides some guidance as to what information is important to vulnerability management decisions and the organization should start capturing, storing, and managing.

Development Methodology

For this tabletop refinement, we could not select a mathematically representative set of CVEs. The goal was to select a handful of CVEs that would cover diverse types of vulnerabilities. The CVEs that we used for our tabletop exercises are CVE-2017-8083, CVE-2019-2712, CVE-2014-5570, and CVE-2017-5753.

We discussed each one from the perspective of supplier and deployer. We evaluated CVE-2017-8083 twice because our understanding and descriptions had changed materially after the first three CVEs (six evaluation exercises). After we were satisfied that the decision trees were clearly defined and captured our intentions, we began the formal evaluation of the draft trees, which we describe in the next section.

Guidance on Communicating Results

There are many aspects of SSVC that two parties might want to communicate. Not every stakeholder will use the decision points to make comparable decisions. Suppliers and deployers make interdependent decisions, but the actions of one group are not strictly dependent on the other. Recall that one reason for this is that SSVC is about prioritizing a vulnerability response action in general, not specifically applying a patch that a supplier produced. Coordinators are particularly interested in facilitating communication because that is their core function. This section handles three aspects of this challenge: formats for communicating SSVC, how to handle partial or incomplete information, and how to handle information that may change over time.

This section is about communicating SSVC information about a specific vulnerability. Any stakeholder making a decision on allocating effort should have a decision tree with it's decision points and possible values specified already. Representation choices and Tree Construction and Customization Guidance discussed how SSVC uses a text file as the canonical form of a decision tree; the example trees can be found in SSVC/data. This section discusses the situation where one stakeholder, usually a supplier or coordinator, wants to communicate some information about a specific vulnerability to other stakeholders or constituents.

Communication Formats

We recommend two structured communication formats, abbreviated and full. The goal of the abbreviated format is to fill a need for providing identifying information about a vulnerability or decision in charts, graphs, and tables. Therefore, the abbreviated format is not designed to stand alone. The goal of the full format is to capture all the context and details about a decision or work item in a clear and machine-readable way.

Abbreviated Format

SSVC abbreviated form borrows directly from the CVSS "vector string" notation. Since the purpose of the abbreviated form is to provide labels for charts and graphics, it does not stand alone. The basic format for SSVC is:

```
SSVC/(version)/(decision point):(value)[/decision point:value[/decision point:value[...]]][/time]/
```

Where `version` is `v2` if it is based on this current version of the SSVC. The term `decision point` is one or two letters derived from the name of the decision point as follows:

- Start with the decision point name as given in Likely Decision Points and Relevant Data.
- Remove any text in parentheses (and the parentheses themselves).
- Remove the word "Impact" if it's part of the name.
- Create an initialism from remaining title-case words (ignore "of," etc.), taking only the first two words.
- The first letter of the initialism is upper case; if there is a second letter, then it is lower case.

- Verify that the initialism is unique among decision points in the version of SSVC. If two initialisms collide, sort their source names equivalent to `LC_ALL=C sort`. The name that sorts first keeps the initialism for which there is a collision. Set the second letter of the initialism to the first character in the name that resolves the collision. If the names were `Threat` and `Threshold`, `T` would be `Threat` and `Ts` would be `Threshold`. We make an effort to design SSVC without such collisions.

For example, *Technical Impact* becomes `T` and *Public Safety Impact* becomes `Ps`.

The term `value` is a statement of the value or possible values of the decision point that precedes it and to which it is connected by a colon (`:`). Similar to `decision point`, `value` should be made up of one or two letters derived from the name of the decision value in the section for its associated decision point. For example MEF support crippled becomes `Ms` and efficient becomes `E`. The process is the same as above except that labels for a `value` do not need to be unique to the SSVC version, just unique to the associated `decision point`.

The character / separates decision-point:value pairs. As many pairs should be provided in the abbreviated form as are required to communicate the desired information about the vulnerability or work item. A vector must contain at least one decision-point:value pair. The ordering of the pairs should be sorted alphabetically from A to Z by the ASCII characters representing the decision points. A trailing / is used to close the string.

The vector is not tied to a specific decision tree. It is meant to communicate information in a condensed form. If priority labels (*defer*, etc.) are connected to a vector, then the decision tree used to reach those decisions should generally be noted. However, for complex communication, machine-to-machine communication, or long-term storage of SSVC data, the JSON format and schema should be used.

The optional parameter `time` is the time in seconds since the UNIX epoch that the SSVC information was collected or last checked for freshness and accuracy.

Based on this, an example string could be:

```
SSVCv2/Ps:Nm/T:T/U:E/1605040000/
```

For a vulnerability with no or minor *Public Safety Impact*, total *Technical Impact*, and efficient *Utility*, which was evaluated on Nov 10, 2020.

While these abbreviated format vectors can be uniquely produced based on a properly formatted JSON object, going from abbreviated form to JSON is not supported. Therefore, JSON is the preferred storage and transmission method.

Full JSON format

For a more robust, self-contained, machine-readable, we provide JSON schemas. The provision schema is equivalent to a decision tree and documents the full set of logical statements that a stakeholder uses to make decisions. The computed schema expresses a set of information about a work item or vulnerability at a point in time. A computed schema should identify the provision schema used, so the options from which the information was computed are specified.

Each element of `choices` should be an object that is a key-value pair of `decision point:value`, where the term `decision point` is a string derived from the name of the decision point as follows:

- Start with the decision point name as given in Likely Decision Points and Relevant Data.

- Remove any text in parentheses (and the parentheses themselves).
- Remove colon characters, if any (`:`).
- Convert the string to lower camel case by lowercasing the string, capitalizing any letter after a space, and removing all spaces.

The `value` term is derived the same way as `decision point` except start with the value name as given in the relevant decision point subsection of Likely Decision Points and Relevant Data.

Partial or Incomplete Information

What an analyst knows about a vulnerability may not be complete. However, the vulnerability management community may still benefit from partial information. In particular, suppliers and coordinators who might not know everything a deployer knows can still provide benefit to deployers by sharing what partial information they do know. A second benefit to providing methods for communicating partial information is the reduction of bottlenecks or barriers to information exchange. A timely partial warning is better than a complete warning that is too late.

The basic guidance is that the analyst should communicate all of the vulnerability's possible states, to the best of the analyst's knowledge. If the analyst knows nothing, all states are possible. For example, *Utility* may be laborious, efficient, or super effective. In abbreviated form, write this as `U:LESe`. Since a capital letter always indicates a new value, this is unambiguous.

The reason a stakeholder might publish something like `U:LESe` is that it expresses that the analyst thought about *Utility* but does not have anything to communicate. A stakeholder might have information to communicate about some decision points but not others. If SSVC uses this format to list the values that are in play for a particular vulnerability, there is no need for a special "I don't know" marker.

The merit in this "list all values" approach emerges when the stakeholder knows that the value for a decision point may be A or B, but not C. For example, say the analyst knows that *Value Density* is diffuse but does not know the value for *Automatability. Then the analyst can usefully restrict *Utility* to one of laborious or efficient. In abbreviated form, write this as `U:LE`. As discussed below, information can change over time. Partial information may be, but is not required to be, sharpened over time into a precise value for the decision point.

Information Changes Over Time

Vulnerability management decisions are dynamic, and may change over time as the available information changes. Information changes are one reason why SSVC decisions should always be timestamped. SSVC decision points have different temporal properties. Some, such as *Utility*, are mostly static. For *Utility* to change, the market penetration or deployment norms of a vulnerable component would have to meaningfully change. Some, such as *State of Exploitation*, may change quickly but only in one direction.

Both of these examples are out of the direct control of the vulnerability manager. Some, such as *Exposure*, change mostly due to actions taken by the organization managing the vulnerable component. If the actor who can usually trigger a relevant change is the organization using SSVC, then it is relatively straightforward to know when to update the SSVC decision. That is, the organization should reevaluate the decision when they make a relevant change. For those decision points that are about topics outside the control of the organization using SSVC, then the organization should occasionally poll their information sources for changes. The cadence or rate of polls is different for each decision point, based on the expected rate of change.

We expect that updating information over time will be most useful where the evidence-gathering process can be automated. Organizations that have mature asset management systems will also find this update process more efficient than those that do not. For an organization without a mature asset management system, we would recommend putting organizational resources into maturing that function before putting effort into regular updates to vulnerability prioritization decision points.

The following decision points are usually out of the control of the organization running SSVC. As an initial heuristic, we suggest the associated polling frequency for each. These frequencies can be customized, as the update frequency is directly related to the organization's tolerance for the risk that the information is out of date. As discussed in Tree Construction and Customization Guidance, risk tolerance is unique to each organization. Risk tolerance and risk appetite are primarily reflected in the priority labels (that is, decisions) encoded in the SSVC decision tree, but information polling frequency is also a risk tolerance decision and each organization may choose different time values.

- *State of Exploitation*: every 1 day
- *Technical Impact*: never (should be static per vulnerability)
- *Utility*: every 6 months
- *Public Safety Impact*: every 1 year

The following decision points are usually in the control of the organization running SSVC and should be reevaluated when a relevant change is made or during annual reviews of assets.

- *Situated Safety Impact*
- *Mission Impact*
- *System Exposure*

If SSVC information is all timestamped appropriately (as discussed earlier in this section), then an analyst can compare the timestamp to the current date and determine whether information is considered stale. The above rates are heuristic suggestions, and organizations may choose different ones. Any public repository of vulnerability information should keep a change log of when values change for each decision point, for each vulnerability. Vulnerability response analysts should keep such change logs as well. Similar to logging practices recommended for incident response (Cichonski et al. 2012), such practices make the process less error-prone and facilitate after-action reviews.

Version 2 Changelog

This section summarizes the changes between SSVC version 2 and SSVC version 1.1 as published at the Workshop on the Ecnomics of Information Security (WEIS 2020). The details of what changes were made can be viewed on the SSVC GitHub issues closed under the SSVC v2 Development project. We addressed about 60 issues. About 10 issues identified "bugs" or errors in version 1.1. About 20 issues improved documentation of tools or improved the clarity of document text. The remaining 30 issues were focused on enhancing SSVC based on feedback received on version 1, though several of the bug fixes and documentation improvements also provided improvements. This section focuses on changes that provided enhancements.

Coordinator stakeholder

Version 1 only considered two stakeholders: those who make software, and those who use information systems. Version 2 introduces a coordinator stakeholder and two distinct decisions for that stakeholder group: vulnerability intake triage and publication about a vulnerability. These decisions use some existing decision points, but also introduce six new decision points to support coordinators in making these decisions. The coordinator stakeholder is based on CERT/CC's experience coordinating vulnerabilities.

Terminology changes

Some terms have been adjusted to better align with other usage in the field or based on feedback. Therefore, "patch developer" became **supplier** and "patch applier" became **deployer**. These terms in v2 better reflect the stakeholder's relationship to the vulnerable component and also help keep clear that SSVC is about prioritization of work items in vulnerability management, not just patches. We have also generally removed the word patch and instead use the more general "remediation" for a complete fix and "mitigation" for actions that reduce risk but do not remove a vulnerability from a system. "Virulence" was renamed *Automatable* in a effort to be more direct and clear, rather than relying on an epidemiology metaphor. We changed "out-of-band" to **out-of-cycle**.

Some concepts needed to be clarified or added. These changes are a bit more substantive than the above terminology changes, but are similar. For example, we clarified how end-of-life products are prioritized with SSVC. We also clarified in Scope concepts around vulnerability identificatin and disambiguation. Version 2 adopts an explicit definition of **risk** (from ISO Guide 73). We also differentiated between vulnerability risk, or that risk arising from an unmanaged vulnerability in an information system, and change risk, or that risk from modifying or updating an information system to mitigate or remediate a vulnerability. SSVC version 2 focuses on assessing and managing vulnerability risk, not change risk. This stance was not explicit in SSVC version 1.

Improvements to decision points

Version 1 had a decision point for well-being impact that was shared between **supplier** and **deployer** stakeholders. Since these types of stakeholder have access to different information about safety and well-being, Version 2 splits this concept into *Public Safety Impact* and *Situated Safety Impact*. The underlying definition remains largely the same. However, *Public Safety Impact* has fewer output options (it is less granular) in recognition that a supplier or coordinator has less information about the context of deployment than a deployer does.

In addition, based on feedback from SSVC users, the SSVC version 2 recommended applier tree makes use of a combined value for *Mission Impact* and *Situated Safety Impact*. The intuition behind this change is that if a person is going to die OR the organization is going to fail (for example, go bankrupt), then the organization will likely want to act with highest priority. Either situation is sufficient to increase the priority, and there do not appear to be situations where a low *Mission Impact* would mitigate a high *Situated Safety Impact* or vice versa. On the other hand, a low *Utility* or *System Exposure* may mitigate a high mission or well-being impact. So the Version 2 recommended tree is more usable than the Version 1 tree, thanks to these changes.

Tree management and communication tools

The section Tree Construction and Customization Guidance is largely new or revised. We produced new software tools for interacting with SSVC, which are documented in that section. Version 2 adds reasoning behind why a stakeholder might customize a decision tree, what aspects of the tree are best to customize, tools for encoding custom trees in JSON, and scripts for visualizing custom trees.

Similarly, the section on Guidance on Communicating Results is largely new. The section presents both an abbreviated and unabridged format for communicating SSVC information about a vulnerability. This communication may be connected to the formats for communicating a whole decision tree. Version 2 also addresses several other questions about SSVC information management, such as handling information changes over time, partial information, sourcing information for each decision point, and how collection and analysis of SSVC decision points can be automated.

Evaluation of the Draft Trees

We conducted a pilot test on the adequacy of the hypothesized decision trees. This section reports results for SSVC version 1. The method of the pilot test is described in Pilot Methodogy. The study resulted in several changes to the hypothesized trees; we capture those changes and the reason for each of them in Pilot Results. The version 1 supplier and deployer trees included the improvements reported in Improvement Instigated by the Pilot.

Pilot Methodology

The pilot study tested inter-rater agreement of decisions reached. Each author played the role of an analyst in both stakeholder groups (supplying and deploying) for nine vulnerabilities. We selected these nine vulnerabilities based on expert analysis, with the goal that the nine cases cover a useful series of vulnerabilities of interest. Specifically, we selected three vulnerabilities to represent safety-critical cases, three to represent regulated-systems cases, and three to represent general computing cases.

During the pilot, we did not form guidance on how to assess the values of the decision points. However, we did standardize the set of evidence that was taken to be true for the point in time representing the evaluation. Given this static information sheet, we did not synchronize an evaluation process for how to decide whether *Exploitation*, for example, should take the value of *none*, *PoC*, or *active*. We agreed on the descriptions of the decision points and the descriptions of their values. The goal of the pilot was to test the adequacy of those descriptions by evaluating whether the analysts agreed. We improved the decision point descriptions based on the results of the pilot; our changes are documented in Improvement Instigated by the Pilot.

In the design of the pilot, we held constant the information available about the vulnerability. This strategy restricted the analyst to decisions based on the framework given that information. That is, it controlled for differences in information search procedure among the analysts. The information search procedure should be controlled because this pilot was about the framework content, not how people answer questions based on that content. After the framework is more stable, a separate study should be devised that shows how analysts should answer the questions in the framework. The basis for the assessment that information search will be an important aspect in using the evaluation framework is based on our experience while developing the framework. During informal testing, often disagreements about a result involved a disagreement about what the scenario actually was; different information search methods and prior experiences led to different understandings of the scenario. This pilot methodology holds the information and scenario constant to test agreement about the descriptions themselves. This strategy makes constructing a prioritization system more tractable by separating problems in how people search for information from problems in how people make decisions. This paper focuses only on the structure of decision making. Advice about how to search for information about a vulnerability is a separate project that will be part of future work. In some domains, namely exploit availability, we have started that work in parallel.

The structure of the pilot test is as follows. Table 11 provides an example of the information provided to each analyst. The supplier portfolio details use ~~strikeout font~~ because this decision item was removed after the pilot. The decision procedure for each case is as follows: for each analyst, for each vulnerability, for each stakeholder group, do the following.

1. Start at the root node of the relevant decision tree (deployer or supplier).
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.

Table 14: Example of Scenario Information Provided to Analysts (Using CVE-2019-9042 as the Example)

Information Item	Description Provided to Analysts
Use of Cyber-Physical System	Sitemagic's content management system (CMS) seems to be fairly popular among smaller businesses because it starts out with a free plan to use it. Then it gradually has small increments in payment for additional features. Its ease-of-use, good designs, and one-stop-shopping for businesses attracts a fair number of clients. Like any CMS, it "manages the creation and modification of digital content. These systems typically support multiple users in a collaborative environment, allowing document management with different styles of governance and workflows. Usually the content is a website" (Wikipedia, 2019)
State of Exploitation	Appears to be active exploitation of this vulnerability according to NVD. They have linked to the exploit: http://www.iwantacve.cn/index.php/archives/116/.
~~Developer Portfolio Details~~	~~Sitemagic is an open-source project. The only thing the brand name applies to is the CMS, and it does not appear to be part of another open-source umbrella. The project is under active maintenance (i.e., it's not a dead project).~~
Technical Impact of Exploit	An authenticated user can upload a .php file to execute arbitrary code with system privileges.
Scenario Blurb	We are a small business that uses Sitemagic to help run business. Sitemagic handles everything from digital marketing and site design to facilitating the e-commerce transactions of the website. We rely on this website heavily, even though we do have a brick-and-mortar store. Many times, products are not available in-store, but are available online, so we point many customers to our online store.
Deployer Mission	We are a private company that must turn a profit to remain competitive. We want to provide customers with a valuable product at a reasonable price, while still turning a profit to run the business. As we are privately held (and not public), we are free to choose the best growth strategy (that is, we are not legally bound to demonstrate quarterly earnings for shareholders and can take a longer-term view if it makes us competitive).

Information Item	Description Provided to Analysts
Deployment of Affected System	We have deployed this system such that only the web designer Cheryl and the IT admin Sally are allowed to access the CMS as users. They login through a password-protected portal that can be accessed anywhere in the world for remote administration. The CMS publishes content to the web, and that web server and site are publicly available.

This test structure produced a series of lists similar in form to the contents of Table 12. Analysts also noted how much time they spent on each vulnerability in each stakeholder group.

Table 15: Example Documentation of a Single Decision Point

Decision Point	Branch Selected	Supporting Evidence
Exploitation=active	Controlled	The CMS has a limited number of authorized users, and the vulnerability is exploitable only by an authenticated user

We evaluated inter-rater agreement in two stages. In the first stage, each analyst independently documented their decisions. This stage produced 18 sets of decisions (nine vulnerabilities across each of two stakeholder groups) per analyst. In the second stage, we met to discuss decision points where at least one analyst differed from the others. If any analyst changed their decision, they appended the information and evidence they gained during this meeting in the "supporting evidence" value in their documentation. No changes to decisions were forced, and prior decisions were not erased, just amended. After the second stage, we calculated some statistical measures of inter-rater agreement to help guide the analysis of problem areas.

To assess agreement, we calculate Fleiss' kappa, both for the value in the leaf node reached for each case and every decision in a case (Fleiss and Cohen 1973). Evaluating individual decisions is complicated slightly because the different paths through the tree mean that a different number of analysts may have evaluated certain items, and Fleiss' kappa requires a static number of raters. "Leaf node reached" is described to pick out the specific path through the tree the analyst selected and to treat that as a label holistically. Measuring agreement based on the path has the drawback that ostensibly similar paths (for example, paths that agree on 3 of 4 decisions) are treated as no more similar than paths that agree on 0 of 4 decisions. The two measures of agreement (per decision and per path) are complementary, so we report both.

Pilot participant details

The pilot participants are the five authors plus one analyst who had not seen the draft system before participating. Five of the six participants had spent at least one year as professional vulnerability analysts prior to the pilot (Spring was the exception). Three of the participants had at least ten years of experience each. The participants experience is primarily as coordinators at the CERT® Coordination Center. On the one hand, this is a different perspective than either suppliers or deployers; on the other, the coordinator role is an information broker that often interacts with these perspectives (Householder, Wassermann, et al. 2020, sec. 3).

These participant demographics limit the generalizability of the results of the pilot. Even though the results cannot be systematically generalized to other analysts, there are at least three benefits to conducting the pilot among this limited demographic. First, it should surface any material tacit disagreements about term usage among the authors. Tacit agreements that are not explained in the text likely survive the pilot study without being acknowledged, but places where the authors tacitly disagreed should be surfaced. We found this to be the case; Improvement Instigated by the Pilot documents these results. Second, the pilot provides a case study that demonstrate SSVC is at least possible for some small group of analysts to use. This achievement is not large, but it is a first step. Third, the pilot provides a proof of concept method and metric that any vulnerability prioritization method could use to examine usability for analysts more generally. While the effect of education on vulnerability assessment with CVSS has been tested (Allodi et al. 2018), we are not aware of any current vulnerability prioritization method that tests usability or agreement among analysts as part of the development process. Future work on SSVC as well as further development of other prioritization methods can benefit from using the method described in the pilot. Future instances should use more representative participant demographics.

Vulnerabilities used as examples

The vulnerabilities used as case studies are as follows. All quotes are from the National Vulnerability Database (NVD) and are illustrative of the vulnerability; however, during the study each vulnerability was evaluated according to information analogous to that in Table 11.

Safety-Critical Cases

- CVE-2015-5374: "Vulnerability ... in [Siemens] Firmware variant PROFINET IO for EN100 Ethernet module... Specially crafted packets sent to port 50000/UDP could cause a denial-of-service of the affected device..."
- CVE-2014-0751: "Directory traversal vulnerability in ... GE Intelligent Platforms Proficy HMI/SCADA - CIMPLICITY before 8.2 SIM 24, and Proficy Process Systems with CIMPLICITY, allows remote attackers to execute arbitrary code via a crafted message to TCP port 10212, aka ZDI-CAN-1623."
- CVE-2015-1014: "A successful exploit of these vulnerabilities requires the local user to load a crafted DLL file in the system directory on servers running Schneider Electric OFS v3.5 with version v7.40 of SCADA Expert Vijeo Citect/CitectSCADA, OFS v3.5 with version v7.30 of Vijeo Citect/CitectSCADA, and OFS v3.5 with version v7.20 of Vijeo Citect/CitectSCADA. If the application attempts to open that file, the application could crash or allow the attacker to execute arbitrary code."

Regulated Systems Cases

- CVE-2018-14781: "Medtronic insulin pump [specific versions] when paired with a remote controller and having the "easy bolus" and "remote bolus" options enabled (non-default), are vulnerable to a capture-replay attack. An attacker can ... cause an insulin (bolus) delivery."
- CVE-2017-9590: "The State Bank of Waterloo Mobile ... app 3.0.2 ... for iOS does not verify X.509 certificates from SSL servers, which allows man-in-the-middle attackers to spoof servers and obtain sensitive information via a crafted certificate."

- CVE-2017-3183: "Sage XRT Treasury, version 3, fails to properly restrict database access to authorized users, which may enable any authenticated user to gain full access to privileged database functions. Sage XRT Treasury is a business finance management application. . . . "

General Computing Cases

- CVE-2019-2691: "Vulnerability in the MySQL Server component of Oracle MySQL (subcomponent: Server: Security: Roles). Supported versions that are affected are 8.0.15 and prior. Easily exploitable vulnerability allows high privileged attacker with network access via multiple protocols to . . . complete DoS of MySQL Server."
- CVE-2019-9042: "[I]n Sitemagic CMS v4.4. . . the user can upload a .php file to execute arbitrary code, as demonstrated by 404.php. This can only occur if the administrator neglects to set FileExtensionFilter and there are untrusted user accounts. . . . "
- CVE-2017-5638: "The Jakarta Multipart parser in Apache Struts 2 2.3.x before 2.3.32 and 2.5.x before 2.5.10.1 has incorrect exception handling and error-message generation during file-upload attempts, which allows remote attackers to execute arbitrary commands via crafted [specific headers], as exploited in the wild in March 2017. . . "

Pilot Results

For each of the nine CVEs, six analysts rated the priority of the vulnerability as both a supplier and deployer. Table 13 summarizes the results by reporting the inter-rater agreement for each decision point. For all measures, agreement (κ) is above zero, which is generally interpreted as some agreement among analysts. Below zero is interpreted as noise or discord. Closer to 1 indicates more or stronger agreement.

How close κ should be to 1 before agreement can be considered strong enough or reliable enough is a matter of some debate. The value certainly depends on the number of options among which analysts select. For those decision points with five options (mission and safety impact), agreement is lowest. Although portfolio value has a higher κ than mission or safety impact, it may not actually have higher agreement because portfolio value only has two options. The results for portfolio value are nearly indistinguishable as far as level of statistical agreement from mission impact and safety impact. The statistical community does not have hard and fast rules for cut lines on adequate agreement. We treat κ as a descriptive statistic rather than a test statistic.

Table 13 is encouraging, though not conclusive. $\kappa<0$ is a strong sign of discordance. Although it is unclear how close to 1 is success, $\kappa<0$ would be clear sign of failure. In some ways, these results may be undercounting the agreement for SSVC as presented. These results are for SSVC prior to the improvements documented in Improvement Instigated by the Pilot, which are implemented in SSVC version 1. On the other hand, the participant demographics may inflate the inter-rater agreement based on shared tacit understanding through the process of authorship. The one participant who was not an author surfaced two places where this was the case, but we expect the organizational homogeneity of the participants has inflated the agreement somewhat. The anecdotal feedback from vulnerability managers at several organizations (including VMware (Akbar 2020) and McAfee) is about refinement and tweaks, not gross disagreement. Therefore, while further refinement is necessary, this evidence suggests the results have some transferability to other organizations and are not a total artifact of the participant organization demographics.

Table 16: Inter-Rater Agreement for Decision Points

	Safety Impact	Exploitation	Technical Impact	Portfolio Value	Mission Impact	Exposure	Dev Result	Deployer Result
Fleiss' κ	0.122	0.807	0.679	0.257	0.146	0.480	0.226	0.295
Disagreement range	2,4	1,2	1,1	1,1	2,4	1,2	1,3	2,3

For all decision points, the presumed goal is for κ to be close or equal to 1. The statistics literature has identified some limited cases in which Fleiss' k behaves strangely—for example it is lower than expected when raters are split between 2 of q ratings when q>2 (Falotico and Quatto 2015). This paradox may apply to the safety and mission impact values, in particular. The paradox would bite hardest if the rating for each vulnerability was clustered on the same two values, for example, minor and major. Falotico and Quatto's proposed solution is to permute the columns, which is safe with unordered categorical data. Since the nine vulnerabilities do not have the same answers as each other (that is, the answers are not clustered on the same two values), we happen to avoid the worst of this paradox, but the results for safety impact and mission impact should be interpreted with some care.

This solution identifies another difficulty of Fleiss' kappa, namely that it does not preserve any order; none and catastrophic are considered the same level of disagreement as none and minor. Table 13 displays a sense of the range of disagreement to complement this weakness. This value is the largest distance between rater selections on a single vulnerability out of the maximum possible distance. So, for safety impact, the most two raters disagreed was by two steps (none to major, minor to hazardous, or major to catastrophic) out of the four possible steps (none to catastrophic). The only values of κ that are reliably comparable are those with the same number of options (that is, the same maximum distance). In other cases, closer to 1 is better, but how close is close enough to be considered "good" changes. In all but one case, if raters differed by two steps then there were raters who selected the central option between them. The exception was mission impact for CVE-201814781; it is unclear whether this discrepancy should be localized to a poor test scenario description, or to SSVC's mission impact definition. Given it is an isolated occurrence, we expect the scenario description at least partly.

Nonetheless, κ provides some way to measure improvement on this a conceptual engineering task. The pilot evaluation can be repeated, with more diverse groups of stakeholders after the descriptions have been refined by stakeholder input, to measure fit to this goal. For a standard to be reliably applied across different analyst backgrounds, skill sets, and cultures, a set of decision point descriptions should ideally achieve κ of 1 for each item in multiple studies with diverse participants. Such a high level of agreement would be difficult to achieve, but it would ensure that when two analysts assign a priority with the system that they get the same answer. Such agreement is not the norm with CVSS currently (Allodi et al. 2018).

Table 17: SSVC pilot scores compared with the CVSS base scores for the vulnerabilities provided by NVD.

CVE-ID	Representative SSVC decision values	SSVC recommendation (supplier, deployer)	NVD's CVSS base score
CVE-2014-0751	E:N/T:T/U:L/S:H/X:C/M:C	(Sched, OOC)	7.5 (High) (v2)
CVE-2015-1014	E:N/T:T/U:L/S:J/X:S/M:F	(Sched, Sched)	7.3 (High) (v3.0)

CVE-ID	Representative SSVC decision values	SSVC recommendation (supplier, deployer)	NVD's CVSS base score
CVE-2015-5374	E:A/T:P/U:L/S:H/X:C/M:F	(Immed, Immed)	7.8 (High) (v2)
CVE-2017-3183	E:N/T:T/U:E/S:M/X:C/M:C	(Sched, Sched)	8.8 (High) (v3.0)
CVE-2017-5638	E:A/T:T/U:S/S:M/X:U/M:C	(Immed, OOC)	10.0 (Critical) (v3.0)
CVE-2017-9590	E:P/T:T/U:E/S:M/X:U/M:D	(OOC, Sched)	5.9 (Medium) (v3.0)
CVE-2018-14781	E:P/T:P/U:L/S:H/X:C/M:F	(OOC, OOC)	5.3 (Medium) (v3.0)
CVE-2019-2691	E:N/T:P/U:E/S:M/X:C/M:C	(Sched, Sched)	4.9 (Medium) (v3.0)
CVE-2019-9042	E:A/T:T/U:L/S:N/X:C/M:C	(OOC, Sched)	7.2 (High) (v3.0)

Table 14 presents the mode decision point value for each vulnerability tested, as well as the recommendation that would result from that set based on the recommended decision trees in SSVC version 1. The comparison with the NVD's CVSS base scores mostly confirms that SSVC is prioritizing based on different criteria, as designed. In particular, differences in the state of exploitation and safety impact are suggestive.

Based on these results, we made about ten changes, some bigger than others. We did not execute a new rater agreement experiment with the updated descriptions. The pilot results are encouraging, and we believe it is time to open up a wider community discussion.

Improvements Instigated by the Pilot

The following changes were reflected in the version 1 Section "Decision Trees for Vulnerability Management."

- Technical impact: We clarified that partial/total is decided regarding the system scope definition, which considers a database or a web server program as the "whole" system. Furthermore, "total" also includes any technical impact that exposes authentication credentials to the adversary, if those credentials are to the whole system.
- We added advice for information gathering to answer safety impact and mission impact questions. This change is needed because of the particularly wide variety of background assumptions analysts made that influenced results and agreement.
- We clarified that "MEF failure" refers to any **one** essential function failing, not failure of all of them. We changed most severe mission impact to "mission failure" to better reflect the relationship between MEFs and the organization's mission.
- We removed the "supplier portfolio value" question since it had poor agreement, and there is no clear way to correct it. We replaced this question with *Utility*, which better captures the relevant kinds of value (namely, to the adversary) of the affected component while remaining amenable to pragmatic analysis.
- We clarified that "proof of concept" (see *Exploitation*) includes cases in which existing tooling counts as a PoC. The examples listed are suggestive, not exhaustive.
- We reorganized the decision trees based on which items are easier to gather information for or which ones have a widely verifiable state. This change moved *exploitation* to the first question.

- We changed the decision tree results such that if exposure is "small," then the resulting priority is lower than before the pilot study. That is, "small" exposure has a stronger effect on reducing urgency.

Questions Removed as Ineffective

In this section, we present ideas we tried but rejected for various reasons. We are not presenting this section as the final word on excluding these ideas, but we hope the reasons for excluding them are instructive, will help prevent others from re-inventing the proverbial wheel, and can guide thinking on future work.

Initially, we brainstormed approximately 12 potential decision points, most of which we removed early in our development process through informal testing. These decision points included adversary's strategic benefit of exploiting the vulnerability, state of legal or regulatory obligations, cost of developing remediation, patch distribution readiness, financial losses to customers due to potential exploitation, and business losses to the deployer.

Some of these points left marks on other decision points. The decision point "financial losses of customers" led to an amendment of the definition of *Safety* to include "well-being," such that, for example, bankruptcies of third parties are now a major safety impact. The "business losses to the deployer" decision point is covered as a mission impact insofar as profit is a mission of publicly traded corporations.

Three of the above decision points left no trace on the current system. "State of legal or regulatory obligations," "cost of developing remediation," and "patch distribution readiness" were dropped as either being too vaguely defined, too high level, or otherwise not within the scope of decisions by the defined stakeholders. The remaining decision point, "adversary's strategic benefit of exploiting the vulnerability," transmuted to a different sense of system value. In an attempt to be more concrete and not speculate about adversary motives, we considered a different sense of value: supplier portfolio value.

The only decision point that we have removed following the pilot is developer portfolio value. This notion of value was essentially an over-correction to the flaws identified in the "adversary's strategic benefit of exploiting the vulnerability" decision point. "Supplier portfolio value" was defined as "the value of the affected component as a part of the developer's product portfolio. Value is some combination of importance of a given piece of software, number of deployed instances of the software, and how many people rely on each. The developer may also include lifecycle stage (early development, stable release, decommissioning, etc.) as an aspect of value." It had two possible values: low and high. As Table 13 demonstrates, there was relatively little agreement among the six analysts about how to evaluate this decision point. We replaced this sense of portfolio value with *Utility*, which combines *Value Density* and *Automatability*.

Worked Example

As an example, we will evaluate CVE-2018-14781 step by step from the deployer point of view. The scenario here is that used for the pilot study. This example uses the SSVC version 1 deployer decision tree.

The analyst's first question is related to exploitation. Technically, one could answer the questions in any order; however, exploitation is a good starting point because given an adequately defined search procedure, one can always answer whether it finds an available exploit or proof of concept. The scenario description for the pilot study reads as follows:

- **State of exploitation**: Metasploit and ExploitDB do not return results for this vulnerability. The NVD does not report any active exploitation of this vulnerability.

This information rules out "active" given the (perhaps limited) search procedure. While the search did not produce a precise PoC, based on the description of the vulnerability, it is a fairly standard traffic capture and replay attack that, given access to the transmission medium, should be straightforward to conduct with Wireshark. Therefore, we select the "PoC" branch and then ask about exposure. This considers the (fictional) deployer scenario blurb and the notional deployment of the affected system, as follows.

- **Scenario blurb**: We are a hospital that uses Medtronic devices frequently because of their quality and popularity in the market. We give these devices out to clients who need to monitor and track their insulin intake. If clients need to access data on their device, they can physically connect it to their computer or connect via Bluetooth to an app on their phone for monitoring capabilities. Occasionally, clients who use this device will have a doctor's appointment in which the doctors have machines that can access the device as well to monitor or change settings. It is unknown how secure the doctor's computer that interfaces directly with this insulin pump is. If the doctor's computer is compromised, it potentially means that every device that connects to it is compromised as well. If an update to the insulin pump is required, a client can do this on their own through their computer or app or through a doctor while they are on-site at the hospital.
- **Deployment of affected system**: These pumps are attached directly to the client. If an update is required, the client is permitted to do that through their own computer or app. However, we have not provided them with documentation on properly using their computer or app to securely access their device. This is done for convenience so that if the user needs to change something quickly, they can. They also can also come to us (hospital) for a change in their device's settings for dosage etc. The doctor's computer that directly handles interfacing with these devices is only connected to the intranet for the purpose of updating the client's settings on the device. Doctors authenticate with ID badge and password.

System Exposure is less straightforward than *Exploitation*. The option *open* is clearly ruled out. However, it is not clear whether the optional Bluetooth connection between the medical device and a phone app represents *controlled* or *small* exposure. The description does not explicitly handle the capture/replay aspect of the vulnerability. If the only way to exploit the vulnerability is to be within physical transmission range of the device, then that physical constraint argues for exposure being *small*. However, if the client's phone app could be used to capture and replay attack packets, then unless that app is particularly well secured, the answer should be *controlled*. Regardless, the answer is not clear from the supplied information. Furthermore, if this fictional app is specific to the insulin pump, then even if it is not compromised, the attack might use its installation to remotely identify targets. However, since most of the hospital's clients have not installed the app, and for nearly all cases, physical proximity to the device is necessary; therefore, we select *small* and move on to ask about mission impact.

According to the fictional pilot scenario, "Our mission dictates that the first and foremost priority is to contribute to human welfare and to uphold the Hippocratic oath (do no harm)." The continuity of operations planning for a hospital is complex, with many MEFs. However, even from this abstract, it seems clear that "do no harm" is at risk due to this vulnerability. A mission essential function to that mission is each of the various medical devices works as expected, or at least if a device fails, it cannot actively be used to inflict harm. Unsolicited insulin delivery would mean that MEF "fails for a period of time longer than acceptable," matching the description of MEF failure. The question is then whether the whole mission fails, which does not seem to be the case. The recovery of MEF functioning is not affected, and most MEFs (the emergency services, surgery, oncology, administration, etc.) would be unaffected. Therefore, we select *MEF failure* and move on to ask about safety impact.

This particular pilot study used SSVC v1. In the suggested deployer tree for SSVC v2, mission and safety impact would be used to calculate the overall *Human Impact*, and *Utility* would need to be answered as well. Conducting further studies with the recommended v2 Deployer tree remains future work. In the pilot study, this information is conveyed as follows:

- **Use of the cyber-physical system**: Insulin pumps are used to regulate blood glucose levels in diabetics. Diabetes is extremely common in the US. Misregulation of glucose can cause a variety of problems. Minor misregulation causes confusion or difficulty concentrating. Long-term minor mismanagement causes weigh management issues and blindness. Severe acute mismanagement can lead unconsciousness in a matter of minutes and death in a matter of hours. The impacted insulin pumps have a local (on-patient) wireless control, so wires to the pump do not have to be connected to the patient's control of the system, making the system lighter and less prone to be ripped out.

The closest match to "death in a matter of hours" would be *hazardous* because that description reads "serious or fatal injuries, where fatalities are plausibly preventable via emergency services or other measures." Depending on the details of the hospital's contingency plans and its monitoring of their patients, the *Safety Impact* could be *catastrophic*. If there is no way to tell whether the insulin pumps are misbehaving, for example, then exploitation could go on for some time, leading to a *catastrophic Safety Impact*. The pilot information is inadequate in this regard, which is the likely source of disagreement about *Safety Impact* in Table 13. For the purposes of this example, imagine that after gathering that information, the monitoring situation is adequate, and select *hazardous*. Therefore, mitigate this vulnerability *out-of-cycle*, meaning that it should be addressed quickly, ahead of the usual update and patch cycle.

Related Vulnerability Management Systems

There are several other bodies of work that are used in practice to assist vulnerability managers in making decisions. Three relevant systems are CVSS (CVSS SIG 2019), EPSS (Jacobs et al. 2019), and Tenable's Vulnerability Priority Rating (VPR). There are other systems derived from CVSS, such as RVSS for robots (Vilches et al. 2018) and MITRE's effort to adapt CVSS to medical devices (Chase and Coley 2019). There are also other nascent efforts to automate aspects of the decision making process, such as vPrioritizer. This section discusses the relationship between these various systems and SSVC.

CVSS

CVSS v3.1 has three metric groups: base, environmental, and temporal. The metrics in the base group are all required, and are the only required metrics. In connection with this design, CVSS base scores and base metrics are far and away the most commonly used and communicated. A CVSS base score has two parts: the exploitability metrics and the impact metrics. Each of these are echoed or reflected in aspects of SSVC, though the breadth of topics considered by SSVC is wider than CVSS v3.1.

How CVSS is used matters. Using just the base scores, which are "the intrinsic characteristics of a vulnerability that are constant over time and across user environments," as a stand-alone prioritization method is not recommended (CVSS SIG 2019). Two examples of this include the U.S. government (see Scarfone et al. 2008, 7–4; Souppaya and Scarfone 2013, 4; and Cybersecurity and Infrastructure Security

Agency 2015) and the global payment card industry (PCI Security Standards Council 2017) where both have defined such misuse as expected practice in their vulnerability management requirements. CVSS scores have a complex relationship with patch deployment in situations where it is not mandated, at least in an ICS context (Wang et al. 2017).

CVSS has struggled to adapt to other stakeholder contexts. Various stakeholder groups have expressed dissatisfaction by making new versions of CVSS, such as medical devices (Chase and Coley 2019), robotics (Vilches et al. 2018), and industrial systems (Figueroa-Lorenzo, Añorga, and Arrizabalaga 2020). In these three examples, the modifications tend to add complexity to CVSS by adding metrics. Product vendors have varying degrees of adaptation of CVSS for development prioritization, including but not limited to Red Hat, Microsoft, and Cisco. The vendors codify CVSS's recommended qualitative severity rankings in different ways, and Red Hat and Microsoft make the user interaction base metric more important.

Exploitability metrics (Base metric group)

The four metrics in this group are Attack Vector, Attack Complexity, Privileges Required, and User Interaction. This considerations may likely be involved in the *Automatability* decision point. If Attack Vector = Network and Privileges Required = None, then the delivery phase of the kill chain is likely to be automatable. Attack Vector may also be correlated with the *Exposure* decision point. Attack Complexity may influence how long it may take an adversary to craft an automated exploit, but *Automatability* only asks whether exploitation can be automated, not how difficult it was. However, Attack Complexity may influence the weaponization phase of the kill chain. User Interaction does not cleanly map to a decision point. In general, SSVC does not care whether a human is involved in exploitation of the vulnerability or not. Some human interaction is for all intents and purposes automatable by attackers: most people click on links in emails as part of their normal processes. In most such situations, user interaction does not present a firm barrier to automatability; it presents a stochastic barrier. *Automatability* is written to just consider firm barriers to automation.

Automatability includes considerations that are not included in the exploitability metrics. Most notably the concept of vulnerability chaining is addressed in *Automatability* but not addressed anywhere in CVSS. *Automatability* is also outcomes focused. A vulnerability is evaluated based on an observable outcome of whether the first four steps of the kill chain can be automated for it. A proof of automation in a relevant environment is an objective evaluation of the score in a way that cannot be provided for some CVSS elements, such as Attack Complexity.

Impact metrics (Base metric group)

The metrics in this group are Confidentiality, Integrity, and Availability. There is also a loosely associated Scope metric. The CIA impact metrics are directly handled by *Technical Impact*.

Scope is a difficult CVSS metric to categorize. The specification describes it as "whether a vulnerability in one vulnerable component impacts resources in components beyond its security scope" (CVSS SIG 2019). This is a fuzzy concept. SSVC better describes this concept by breaking it down into component parts. The impact of exploitation of the vulnerable component on other components is covered under *Mission Impact*, public and situated *Well-being Impact*, and the stakeholder-specific nature where SSVC is tailored to stakeholder concerns. CVSS addresses some definitions of the scope of CVSS as a whole under the Scope metric definition. In SSVC, these definitions are in the Scope section.

Temporal metric groups

The temporal metric group primarily contains the Exploit Code Maturity metric. This metric expresses a concept similar to *Exploitation*. The main difference is that *Exploitation* is not optional in SSVC and that SSVC accounts for the observation that most vulnerabilities with CVE-IDs do not have public exploit code (Householder, Chrabaszcz, et al. 2020) and are not actively exploited Jacobs et al. (2019).

Environmental metric group

The environmental metric group allows a consumer of a CVSS base score to change it based on their environment. CVSS needs this functionality because the organizations that produce CVSS scores tend to be what SSVC calls **suppliers** and consumers of CVSS scores are what SSVC calls **deployers**. These two stakeholder groups have a variety of natural differences, which is why SSVC treats them separately. SSVC does not have such customization as a bolt-on optional metric group because SSVC is stakeholder-specific by design.

EPSS

EPSS is an "effort for predicting when software vulnerabilities will be exploited." EPSS is currently based on a machine-learning classifier and proprietary IDS alert data from Kenna Security. While the group has made an effort to make the ML classifier transparent, ML classifiers are not able to provide an intelligible, human-accessible explanation for their behavior (Spring et al. 2019). The use of proprietary training data makes the system less transparent.

EPSS could be used to inform the *Exploitation* decision point. Currently, *Exploitation* focuses on the observable state of the world at the time of the SSVC decision. EPSS is about predicting if a transition will occur from the SSVC state of *none* to *active*. A sufficiently high EPSS score could therefore be used as an additional criterion for scoring a vulnerability as *active* even when there is no observed active exploitation.

VPR

VPR is a prioritization product sold by Tenable. VPR determines the severity level of a vulnerability based on "technical impact and threat." Just as *Technical Impact* in SSVC, technical impact in VPR tracks the CVSSv3 impact metrics in the base metric group. The VPR threat component is about recent and future threat activity; it is comparable to *Exploitation* if EPSS were added to *Exploitation*.

VPR is therefore essentially a subset of SSVC. VPR is stylistically methodologically quite different from SSVC. VPR is based on machine learning models and proprietary data, so the results are totally opaque. There is no ability to coherently and transparently customize the VPR system. Such customization is a central feature of SSVC, as described in Tree Construction and Customization Guidance.

CVSS spin offs

Attempts to tailor CVSS to specific stakeholder groups, such as robotics or medical devices, are are perhaps the biggest single reason we created SSVC. CVSS is one-size-fits-all by design. These customization efforts struggle with adapting CVSS because it was not designed to be adaptable to different stakeholder considerations. The SSVC section Tree Construction and Customization Guidance explains how stakeholders or stakeholder communities can adapt SSVC in a reliable way that still promotes repeatability and communication.

vPrioritizer

vPrioritizer is an open-source project that attempts to integrate asset management and vulnerablity prioritization. The software is mostly the asset management aspects. It currently includes CVSS base scores as the de facto vulnerability prioritization method; however, fundamentally the system is agnostic to prioritization method. vPrioritizer is an example of a product that is closely associated with vulnerability prioritization, but is not directly about the prioritization method. In that sense, it is compatible with any of methods mentioned above or SSVC. However, SSVC would be better suited to address vPrioritizer's broad spectrum asset management data. For example, vPrioritizer aims to collect data points on topics such as asset significance. Asset significance could be expressed through the SSVC decision points of *Mission Impact* and situated *Well-being Impact*, but it does not have a ready expression in CVSS, EPSS, or VPR.

SSVC using Current Information Sources

Some SSVC decision points can be informed or answered by currently available information feeds or sources. These include *Exploitation*, *System Exposure*, *Technical Impact*, and *Public Safety Impact*. This section provides an overview of some options; we cannot claim it is exhaustive. Each decision point has a subsection for `Gathering Information About` it. These sections provide suggestions that would also contribute to creating or honing information feeds. However, if there is a category of information source we have not captured, please create an issue on the SSVC GitHub page explaining it and what decision point it informs.

Various vendors provide paid feeds of vulnerabilities that are currently exploited by attacker groups. Any of these could be used to indicate that *active* is true for a vulnerability. Although the lists are all different, we expect they are all valid information sources; the difficulty is matching a list's scope and vantage with a compatible scope and vantage of the consumer. We are not aware of a comparative study of the different lists of active exploits; however, we expect they have similar properties to block lists of network touchpoints (Metcalf and Spring 2015) and malware (Kührer, Rossow, and Holz 2014). Namely, each list has a different view and vantage on the problem, which makes them appear to be different, but each list accurately represents its particular vantage at a point in time.

System Exposure could be informed by the various scanning platforms such as Shodan and Shadowserver. A service on a device should be scored as *open* if such a general purpose Internet scan finds that the service responds. Such scans do not find all *open* systems, but any system they find should be considered *open*. Scanning software, such as the open-source tool Nessus, could be used to scan for connectivity inside an organization to catalogue what devices should be scored *controlled* if, say, the scan finds them on an internal network where devices regularly connect to the Internet.

Some information sources that were not designed with SSVC in mind, but can be adapted to work with it. Three prominent examples are CVSS impact base metrics, CWE, and CPE.

Technical Impact is directly related to the CVSS impact metric group. However, this metric group cannot be directly mapped to *Technical Impact* in CVSSv3 because of the Scope metric. *Technical Impact* is only about adversary control of the vulnerable component. If the CVSSv3 value of "Scope" is "Changed," then the impact metrics are the maximum of the impact on the vulnerable component and other components in the environment. If confidentiality, integrity, and availability metrics are all "high" then *Technical Impact* is *total*, as long as the impact metrics in CVSS are clearly about just the vulnerable component. However, the other values of the CVSSv3 impact metrics cannot be mapped directly to *partial* because of

CVSSv3.1 scoring guidance. Namely, "only the increase in access, privileges gained, or other negative outcome as a result of successful exploitation should be considered" (CVSS SIG 2019). The example given is that if an attacker already has read access, but gains all other access through the exploit, then read access didn't change and the confidentiality metric score should be "None" . However, in this case, SSVC would expect the decision point to be evaluated as *total* because as a result of the exploit the attacker gains total control of the device, even though they started with partial control.

As mentioned in the discussion of *Exploitation*, CWE could be used to inform one of the conditions that satisfy *proof of concept*. For some classes of vulnerabilities, the proof of concept is well known because the method of exploitation is already part of open-source tools. For example, on-path attacker scenarios for intercepting TLS certificates. These scenarios are a cluster of related vulnerabilities. Since CWE classifies clusters of related vulnerabilities, the community could likely curate a list of CWE-IDs for which this condition of well known exploit technique is satisfied. Once that list were curated, it could be used to automatically populate a CVE-ID as *proof of concept* if the CWE-ID of which it is an instance is on the list. Such a check could not be exhaustive, since there are other conditions that satisfy *proof of concept*. If paired with automatic searches for exploit code in public repositories, these checks would cover many scenarios. If paired with active exploitation feeds discussed above, then the value of *Exploitation* could be determined almost entirely from available information without direct analyst involvement at each organization.

CPE could possibly be curated into a list of representative *Public Safety Impact* values for each platform or product. The *Situated Safety Impact* would be too specific for a classification as broad as CPE. But it might work for *Public Safety Impact*, since it is concerned with a more general assessment of usual use of a component. Creating a mapping between CPE and *Public Safety Impact* could be a community effort to associate a value with each CPE entry, or an organization might label a fragment of the CPE data with *Public Safety Impact* based on the platforms that the supplier needs information about most often.

Potential Future Information Feeds

So far, we have identified information sources that can support scalable decision making for most decision points. Some sources, such as CWE or existing asset management solutions, would require a little bit of connective glue to support SSVC, but not too much. The SSVC decision point that we have not identified an information source for is *Utility*. *Utility* is composed of *Automatable* and *Value Density*, so the question is what a sort of feed could support each of those decision points.

A feed is plausible for both of these decision points. The values for *Automatable* and *Value Density* are both about the relationship between a vulnerability, the attacker community, and the aggregate state of systems connected to the Internet. While that is a broad analysis frame, it means that any community that shares a similar set of adversaries and a similar region of the Internet can share the same response to both decision points. An organization in the People's Republic of China may have a different view than an organization in the United States, but most organizations within each region should should have close enough to the same view to share values for *Automatable* and *Value Density*. These factors suggest a market for an information feed about these decision points is a viable possibility.

At this point, it is not clear that an algorithm or search process could be designed to automate scoring *Automatable* and *Value Density*. It would be a complex natural language processing task. Perhaps a machine learning system could be designed to suggest values. But more likely, if there is a market for this information, a few analysts could be paid to score vulnerabilities on these values for the community.

Supporting such analysts with further automation could proceed by small incremental improvements. For example, perhaps information about whether the Reconnaissance step in the kill chain is *Automatable* or not could be automatically gathered from Internet scanning firms such as Shodan or Shadowserver. This wouldn't make a determination for an analyst, but would be a step towards automatic assessment of the decision point.

Future Work

We intend SSVC to offer a workable baseline from which to improve and refine a vulnerability-prioritization methodology. We are working to improve SSVC. Several of the future work items in this section have issues associated with them on the SSVC GitHub page (https://github.com/CERTCC/SSVC/issues), which is a good place to go to check on progress or help. Plans for future work focus on further requirements gathering, analysis of types of risk, and further testing of the reliability of the decision process.

Requirements Gathering via Sociological Research

The community should know what users of a vulnerability prioritization system want. To explore their needs, it's important to understand how people actually use CVSS and what they think it tells them. In general, such empirical, grounded evidence about what practitioners and decision makers want from vulnerability scoring is lacking. We have based this paper's methodology on multiple decades of professional experience and myriad informal conversations with practitioners. Such evidence is not a bad place to start, but it does not lend itself to examination and validation by others. The purpose of understanding practitioner expectations is to inform what a vulnerability-prioritization methodology should actually provide by matching it to what people want or expect. The method this future work should take is long-form, structured interviews. We do not expect anyone to have access to enough consumers of CVSS to get statistically valid results out of a short survey, nor to pilot a long survey.

Coordinators in particular are likely to be heterogeneous. While the FIRST service frameworks for PSIRTs and CSIRTs differentiate two broad classes of coordinators, we have focused on CSIRTs here. PSIRTs may have somewhat different concerns. Investigating the extent to which SSVC should be customized for this group is future work as well.

Types of Risks

SSVC estimates the relative risk created by a vulnerability in an information system. The priority of acting to mitigate or remediate a vulnerability goes up as this vulnerability risk goes up. SSVC does not currently take into account the change risk due to applying a mitigation or remediation.

One way to view what SSVC currently provides is that it tells you how urgently a stakeholder should analyze overall risk due to a vulnerability. For all but the most dire vulnerabilities, what the stakeholder chooses to do may include accepting the vulnerability risk because the change risk or other costs of mitigation or remediation are too high. Future work should attempt to provide a method for evaluating change risk or cost relative to vulnerability risk.

Tree Construction and Customization Guidance discusses how the prioritization labels in an SSVC tree reflect risk appetite or risk tolerance. Specifically, these reflect vulnerability risk appetite. Appetite for vulnerability risk may be negatively correlated with change risk; future work could explore this relationship. Furthermore, future work could examine suggested practices for connecting tree customization to risk management.

Reasoning Steps Forward states the scope of SSVC analysis is "consider credible effects based on known use cases of the software system as a part of cyber-physical systems." The unit of prioritization in SSVC should be work items. For deployers, a work item is often applying a patch that addresses multiple vulnerabilities. The "credible effects" to consider are those of all vulnerabilities remediated by the patch. How exactly to aggregate these different effects is not currently specified except to say that the unit of analysis is the whole work item. Future work should provide some examples of how this holistic analysis of multiple vulnerabilities remediated in one patch should be conducted.

Further Decision Tree Testing

More testing with diverse analysts is necessary before the decision trees are reliable. In this context, **reliable** means that two analysts, given the same vulnerability description and decision process description, will reach the same decision. Such reliability is important if scores and priorities are going to be useful. If they are not reliable, they will vary widely over time and among analysts. Such variability makes it impossible to tell whether a difference in scores is really due to one vulnerability being higher priority than other.

The SSVC v1 pilot study provides a methodology for measuring and evaluating reliability of the decision process description based on the agreement measure κ. This study methodology should be repeated with different analyst groups, from different sectors and with different experience, using the results to make changes in the decision process description until the agreement measure is adequately close to 1.

Internationalization and localization of SSVC will also need to be considered and tested in future work. It's not clear how best to consider translating SSVC decision points, if at all. And at a very practical level, the Abbreviated Format would have to define a new algorithm for creating initialisms that is not dependent an an alphabet for languages based on syllabaries or ideograms. But a more actionable item of future work would be to include non-native English speakers in future usability studies.

A different approach to testing the *Utility* decision point could be based on Alternative Utility Outputs. Namely, future work could example exploit resale markets and compare the value of exploits to the *Utility* score of the exploited vulnerability. There are some limitations to this approach, since exploit markets target certain adversary groups (such as those with lots of resources) and may not be representative of all adversary types. However, such analysis would provide some information as to whether the definition of *Utility* is reasonable.

Limitations

SSVC has some inherent limits in its approach, which should be understood as tradeoffs. There are other limiting aspects of our implementation, but those have been covered as topics that need improvement and are described in Future Work.

We made two important tradeoffs compared to the current state of the practice. Other systems make different tradeoffs, which may be better or worse depending on the context of use. While these are inherently limitations of SSVC, we do not intend SSVC to be the one and only vulnerability management tool available. Those for whom these limitations are a must-have may be better supported by a different vulnerability management framework.

1. We eliminated numerical scores; this may make some practitioners uncomfortable. We explained the reasons for this in depth, but even though CVSS contains false precision, we still must contend

with the fact that, psychologically, users find that comforting. As this comfort gap may negatively impact adoption, this fact is a limitation. Although it is ungainly, it would be sound to convert the priority outcomes to numbers at the end of the process, if existing processes require it. Which numbers we choose to convert to is immaterial, as long as the ordering is preserved. CVSS has set a precedent that higher numbers are worse, so a scale [1, 2, 3, 4] would work, with *defer* = 1 and *immediate* = 4. However, if it were important to maintain backwards compatibility to the CVSS range zero to ten, we could just as well relabel outcomes as [2, 5.5, 8, 9.5] for the midpoints of the current CVSS severity ranges. This is not a calculation of any kind, just an assignment of a label which may make adoption more conventient. Of course, these labels are dangerous, as they may be misused as numbers. Therefore, we prefer the use *defer*, *scheduled*, etc., as listed in Enumerating Vulnerability Management Actions.

2. We incorporated a wider variety of inputs from contexts beyond the affected component. Some organizations are not prepared or configured to reliably produce such data (e.g., around mission impact or safety impact). There is adequate guidance for how to elicit and curate this type information from various risk management frameworks, including OCTAVE (Caralli et al. 2007). Not every organization is going to have sufficiently mature risk management functions to apply SSVC. This second limitation should be approached with two strategies:
    (a) Organizations should be encouraged and enabled to mature their risk management capabilities
    (b) In the meantime, organizations such as NIST could consider developing default advice. The most practical framing of this approach might be for the NIST NVD to produce scores from the perspective of a new stakeholder—something like "national security" or "public well-being" that is explicitly a sort of default advice for otherwise uninformed organizations that can then explicitly account for national priorities, such as critical infrastructure.

Conclusion

SSVC v2 presents a method for suppliers, deployers, and coordinators to use to prioritize their effort to mitigate vulnerabilities. We have built on SSVC v1 through public presentation and feedback, private consultation, and continued analyst testing. The evaluation process we developed in v1 remains an important part of continued improvement of SSVC, and will be used to continue refinements of SSVC v2. We invite participation and further refinement of the prioritization mechanism from the community as well, such as by posting an issue. We endeavored to be transparent about our process and provide justification for design decisions.

We invite questions, comments, and further community refinement in moving forward with a transparent and justified vulnerability prioritization methodology that is inclusive for the various stakeholders and industries that develop and use information and computer technology.

Acknowledgements

The authors thank the following people for helpful comments on prior drafts: Deana Shick; Will Dormann (CERT/CC); Michel van Eeten and the anonymous WEIS reviewers; Sounil Yu and other attendees at A Conference on Defense (ACoD), Austin TX 2020; Dale Peterson, Ralph Langer, and attendees at S4, Miami FL 2020; Muhammad Akbar and Manish Gaur (VMWare); David Oxley (McAfee); CISA analysts; Jeroen van der Ham; CVSS and EPSS SIG members; and others who wish to remain anonymous.

References

Akbar, Muhammad. 2020. "A Critical First Look at Stakeholder Specific Vulnerability Categorization (SSVC)." March 6, 2020. https://blog.secursive.com/posts/critical-look-stakeholder-specific-vulnerability-categorization-ssvc/.

Allodi, Luca, Marco Cremonini, Fabio Massacci, and Woohyun Shim. 2018. "The Effect of Security Education and Expertise on Security Assessments: The Case of Software Vulnerabilities." In *Workshop on Economics of Information Security*. Innsbruck, Austria.

Allodi, Luca, and Fabio Massacci. 2012. "A Preliminary Analysis of Vulnerability Scores for Attacks in Wild: The EKITS and SYM Datasets." In *Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, 17–24. ACM.

Bano, Shehar, Philipp Richter, Mobin Javed, Srikanth Sundaresan, Zakir Durumeric, Steven J Murdoch, Richard Mortier, and Vern Paxson. 2018. "Scanning the Internet for Liveness." *SIGCOMM Computer Communication Review* 48 (2): 2–9.

Benetis, Vilius, Olivier Caleff, Cristine Hoepers, Angela Horneman, Allen Householder, Klaus-Peter Kossakowski, Art Manion, et al. 2019. "Computer Security Incident Response Team (CSIRT) Services Framework." ver. 2. Cary, NC, USA: FIRST.

Captera. 2019. "IT Asset Management Software." 2019. https://www.capterra.com/it-asset-management-software/.

Caralli, Richard, James Stevens, Lisa Young, and William Wilson. 2007. "Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process." CMU/SEI-2007-TR-012. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. https://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8419.

Cebula, James L, and Lisa R Young. 2010. "A Taxonomy of Operational Cyber Security Risks." CMU/SEI-2010-TN-028. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9395.

Chase, Melissa P, and Steven M Cristey Coley. 2019. "Rubric for Applying CVSS to Medical Devices." 18-2208. McLean, VA, USA: MITRE Corporation.

Cichonski, Paul, Tom Millar, Tim Grance, and Karen Scarfone. 2012. "Computer Security Incident Handling Guide." SP 800-61r2. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards; Technology.

CVE Board. 2020. "CVE Numbering Authority (CNA) Rules." ver. 3.0. Bedford, MA: MITRE. https://cve.mitre.org/cve/cna/rules.html.

CVSS SIG. 2019. "Common Vulnerability Scoring System." version 3.1 r1. Cary, NC, USA: Forum of Incident Response; Security Teams. https://www.first.org/cvss/v3.1/specification-document.

Cybersecurity and Infrastructure Security Agency. 2015. "Critical Vulnerability Mitigation." May 21, 2015. https://cyber.dhs.gov/bod/15-01/.

FAA. 2000. "System Safety Handbook." Washington, DC: US Dept. of Transportation, Federal Aviation Administration. https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/risk_management/ss_handbook/.

Falotico, Rosa, and Piero Quatto. 2015. "Fleiss' Kappa Statistic Without Paradoxes." *Quality & Quantity* 49 (2): 463–70.

Farris, Katheryn A, Ankit Shah, George Cybenko, Rajesh Ganesan, and Sushil Jajodia. 2018. "VULCON: A System for Vulnerability Prioritization, Mitigation, and Management." *Transactions on Privacy and Security* 21 (4): 16.

Fenton, Robert J., ed. 2017. "Federal Continuity Directive 2: Federal Executive Branch Mission Essential Functions and Candidate Primary Mission Essential Functions Identification and Submission Process." US Department of Homeland Security, Federal Emergency Management Agency. https://www.fema.gov/media-library-data/1499702987348-c8eb5e5746bfc5a7a3cb954039df7fc2/FCD-2June132017.pdf.

Figueroa-Lorenzo, Santiago, Javier Añorga, and Saioa Arrizabalaga. 2020. "A Survey of IIoT Protocols: A Measure of Vulnerability Risk Analysis Based on CVSS." *ACM Comput. Surv.* 53 (2). https://doi.org/10.1145/3381038.

Fleiss, Joseph L, and Jacob Cohen. 1973. "The Equivalence of Weighted Kappa and the Intraclass Correlation Coefficient as Measures of Reliability." *Educational and Psychological Measurement* 33 (3): 613–19.

Garfinkel, Simson, and Heather Richter Lipford. 2014. "Usable Security: History, Themes, and Challenges." *Synthesis Lectures on Information Security, Privacy, and Trust* 5 (2): 1–124.

Guido, Dan. 2011. "The Exploit Intelligence Project." iSEC Partners. http://www.trailofbits.com/resources/exploit_intelligence_project_2_slides.pdf.

Householder, Allen D, Jeff Chrabaszcz, Trent Novelly, David Warren, and Jonathan M Spring. 2020. "Historical Analysis of Exploit Availability Timelines." In *Workshop on Cyber Security Experimentation and Test*. Virtual conference: USENIX.

Householder, Allen D, Garret Wassermann, Art Manion, and Christopher King. 2020. "The CERT Guide to Coordinated Vulnerability Disclosure." CMU/SEI-2017-TR-022. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. https://vuls.cert.org/confluence/display/CVD/Executive+Summary.

Howard, Ronald A, and James E Matheson, eds. 1983. *Readings on the Principles and Applications of Decision Analysis: General Collection*. Vol. 1. Strategic Decisions Group.

Hutchins, Eric M, Michael J Cloppert, and Rohan M Amin. 2011. "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains." *Leading Issues in Information Warfare & Security Research* 1: 80.

ISO. 2009. "Risk Management – Vocabulary." 73:2009(en). Geneva, CH: International Organization for Standardization. https://www.iso.org/obp/ui/#iso:std:iso:guide:73:ed-1:v1:en.

Jacobs, Jay, Sasha Romanosky, Benjamin Edwards, Michael Roytman, and Idris Adjerid. 2019. "Exploit Prediction Scoring System (EPSS)." In *Workshop on the Economics of Information Security*. Boston, MA. https://arxiv.org/abs/1908.04856.

Jump, Michelle, and Art Manion. 2019. "Framing Software Component Transparency: Establishing a Common Software Bill of Material (SBOM)." Washington, DC: National Telecommunications; Information Administration.

Kührer, Marc, Christian Rossow, and Thorsten Holz. 2014. "Paint It Black: Evaluating the Effectiveness of Malware Blacklists." In *Recent Advances in Intrusion Detection*, 1–21. LNCS 8688. Gothenburg, Sweden: Springer.

Laube, Stefan, and Rainer Böhme. 2017. "Strategic Aspects of Cyber Risk Information Sharing." *ACM Comput. Surv.* 50 (5). https://doi.org/10.1145/3124398.

Manion, Art, Kazuya Togashi, Joseph B. Kadane, Fumihiko Kousaka, Shawn McCaffrey, Christopher King, Masanori Yamaguchi, and Robert Weiland. 2009. "Effectiveness of the Vulnerability Response Decision Assistance (VRDA) Framework." Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=50301.

Metcalf, Leigh B, and Jonathan M Spring. 2015. "Blacklist Ecosystem Analysis: Spanning Jan 2012 to Jun 2014." In *Workshop on Information Sharing and Collaborative Security*, 13–22. Denver: ACM.

Office of the DoD Chief Information Officer. 2020. "DoD Instruction 8531.01 DoD Vulnerability Management." Washington, DC: Department of Defense. https://fas.org/irp/doddir/dod/i8531_01.pdf.

PCI Security Standards Council. 2017. "Payment Card Industry (PCI) Data Security Standard: Approved Scanning Vendors." ver 3.0. Wakefield, MA, USA. https://www.pcisecuritystandards.org/documents/ASV_Program_Guide_v3.0.pdf.

Pendleton, Marcus, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. 2016. "A Survey on Systems Security Metrics." *ACM Comput. Surv.* 49 (4): 62:1–35.

RTCA, Inc. 2012. "Software Considerations in Airborne Systems and Equipment Certification." DO-178C. Washington, DC: EUROCAE WG-12.

Russell, Stuart J, and Peter Norvig. 2011. *Artificial Intelligence: A Modern Approach*. 3rd ed. Upper Saddle River, NJ: Pearson Education.

Scarfone, Karen, Murugiah Souppaya, Amanda Cody, and Angela Orebaugh. 2008. "Technical Guide to Information Security Testing and Assessment." SP 800-115. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards; Technology.

Simon, Herbert A. 1996. *The Sciences of the Artificial*. 3rd ed. Cambridge, MA: MIT press.

Souppaya, Muragiah, and Karen Scarfone. 2013. "Guide to Enterprise Patch Management Technologies." SP 800-40r3. Gaithersburg, MD: US Dept of Commerce, National Institute of Standards; Technology.

Spring, Jonathan M., Joshua Fallon, April Galyardt, Angela Horneman, Leigh Metcalf, and Ed Stoner. 2019. "Machine Learning in Cybersecurity: A Guide." CMU/SEI-2019-TR-005. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=633583.

Spring, Jonathan M, Eric Hatleback, Allen D Householder, Art Manion, and Deana Shick. 2018. "Towards Improving CVSS." Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Spring, Jonathan M, and Phyllis Illari. 2018. "Building General Knowledge of Mechanisms in Information Security." *Philosophy & Technology* 32 (September): 627–59. https://doi.org/10.1007/s13347-018-0329-z.

———. 2019. "Review of Human Decision-Making During Incident Analysis." *CoRR* abs/1903.10080 (April). http://arxiv.org/abs/1903.10080.

Spring, Jonathan M, Sarah Kern, and Alec Summers. 2015. "Global Adversarial Capability Modeling." In *APWG Symposium on Electronic Crime Research (eCrime)*. Barcelona: IEEE.

Spring, Jonathan M, Tyler Moore, and David Pym. 2017. "Practicing a Science of Security: A Philosophy of Science Perspective." In *New Security Paradigms Workshop*. Santa Cruz, CA, USA. https://tylermoore.utulsa.edu/nspw17.pdf.

Tucker, Brett. 2018. "OCTAVE FORTE and FAIR Connect Cyber Risk Practitioners with the Boardroom." June 2018. https://insights.sei.cmu.edu/insider-threat/2018/06/octave-forte-and-fair-connect-cyber-risk-practitioners-with-the-boardroom.html.

Vilches, V'ictor Mayoral, Endika Gil-Uriarte, Irati Zamalloa Ugarte, Gorka Olalde Mendia, Rodrigo Izquierdo Pisón, Laura Alzola Kirschgens, Asier Bilbao Calvo, Alejandro Hernández Cordero, Lucas Apa, and César Cerrudo. 2018. "Towards an Open Standard for Assessing the Severity of Robot Security Vulnerabilities, the Robot Vulnerability Scoring System (RVSS)." *arXiv Preprint arXiv:1807.10357*.

Wang, Brandon, Xiaoye Li, Leandro P de Aguiar, Daniel S Menasche, and Zubair Shafiq. 2017. "Characterizing and Modeling Patching Practices of Industrial Control Systems." *Measurement and Analysis of Computing Systems* 1 (1): 1–23.

Wiles, Darius, and Dave Dugal, eds. 2019. "Common Vulnerability Scoring System SIG." FIRST. 2019. https://www.first.org/cvss/.

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612
Phone: 412.268.5800 | 888.201.4479
Web: www.sei.cmu.edu
Email: info@sei.cmu.edu