Potential Abuse of Metrics and Economic Value of Metrics

There are many aspects to what constitutes a good metric, which may include factors such as:

- It should be objective and measurable, and facilitate trend analysis (both positive and negative) as well as “apples to apples” comparisons
- It should be economically feasible to gather and use (that is, the cost of obtaining the metric should be exceeded by the management value it provides; if not, it is not worth gathering because it consumes scarce resources that could be put to better economic use (see Economic Value of Metrics)
- It should answer basic questions like (for a vendor) “Are we doing better or worse?"
- It should foster additional questions (e.g., “Why is this number changing?”) That is, metrics should not in general be used to primarily assign blame or find a “throat to choke,” but should lead to additional analysis
- It should help incent the correct behavior (that is, metrics should be used to improve management or lead to better performance and not merely incent “making the numbers look good”)

Potential Abuse of Metrics

The last item in the above list is perhaps a neglected aspect of what constitutes a good metric, but it is critical on order for metrics to be used to *manage better*. A metric that incents the wrong behavior replaces substance with form, and many metrics (especially “single” metrics that are not triangulated or are not part of a group of metrics related to a particular area) are subject to “gaming” or “abuse” in order to make the numbers good. Paradoxically, “making the numbers look good” may result in worse management or worse overall conditions, making the metric not merely inaccurate, but a perverse incentive.

To provide a real-life example of a perverse metric, in one development organization, management ran “open bug reports” every Friday at noon, in order to “measure” how quickly bugs were being closed and absolute numbers of bugs by module. Product managers would routinely “close” open bugs around 11AM on Friday, then reopen them Friday afternoon. (What was accomplished besides wasting everyone’s time “making the numbers look good?”) If people are being measured on one thing, they will manage to it, even if it makes no sense.

Some of the most-desired (and most abused) metrics relate to bug counts and time to fix critical security vulnerabilities. “Numbers of vulnerabilities” as published arguably leads to almost no useful or objective data today since

- The only “apples to apples” comparison is defects per 1000 lines of code, which is a quality metric but not a security one, since security issues have criticality attached which makes, for example, a vulnerability with a CVSS score of 8 more critical to fix in a timely manner than one with a CVSS score of 1.
- No two vendors’ products contain the exact same functionality or same factoring (pricing and bundling of features). A more feature-rich product will likely have more absolute numbers of flaws, including security flaws just like a grass shack will have fewer construction defects than a 400-unit apartment building.
- If a single code change (for example, to a package) fixes three reported vulnerabilities, is it one bug or three that should be counted and “reported?”
- If a bug cannot be fixed at all on old versions (for architectural or backwards compatibility reasons), a vendor may not report it publicly at all (for good security reasons and in the best interests of customers). Which means that issues are underreported (but for a good reason).
- Is “time to fix” measuring the time when any fix is out (that is, when a new version ships that contains the fix already), or when all supported versions have a patch?

“Time to fix” can also be misused as a metric since a good developer being measured on “closing a bug quickly” will do everything possible to close a bug quickly, including denying it is a bug (that is reclassifying it as Not A Bug). Or, which is particularly problematic for a security bug, a developer may only fix the exact manifestation reported and not look at the true underlying cause (e.g., if “CTRL-X” causes the stack to fall over, then changing the code to check for CTRL-X fixes the bug, in theory). However, the true root cause in this case is a failure of correct input validation. In the worst case scenario, the “as manifested” bug is fixed quickly, a single issue patch is put out, which has to be repeated multiple times at significant cost because the root cause of the security bug was not addressed, just “time to fix reported issue.” In the above case, a better metric or an additional one would be whether the security bug is ever reopened due to not being correctly addressed the first time.

Another potentially readily abused metric is “number of reported bugs in a vendor’s software,” absent some normalization. For example, if vendors do not publish their disclosure policy (Do they “self report” bugs they find themselves? On what basis? Severity?) meaningful comparisons are not possible. Unfortunately, we already have instances of vendors competing publicly on “number of reported bugs” where they have explicitly gagged the third party researchers who they have hired to find them and “silent fixed” issues without self-disclosure. To their credit, they have fixed critical issues, but the “numbers” are not meaningful for comparison purposes.

The incentive if numbers are not explained and normalized is for vendors to minimize what they disclose (which may not be in customers’ best interests and is certainly not an objective, unbiased metric). If a vendor finds 80% of security vulnerabilities themselves, then they can “improve” their published numbers by 80% by just stopping any attempt to look for or correct security bugs in their own code base. Obviously, this is not in

anyone’s interests if the goal is actual improved assurance and lower lifecycle security costs.

One way to not only use metrics better but to mitigate “gaming” – that is, managing *to* the numbers instead of *with* them - is to use a combination of measurements to determine how well one is doing. Just as you cannot determine your global position by longitude alone, but only by “triangulation,” a combined metric helps determine where you are relative to the greater universe, and measure overall progress instead of one thing that can and likely will be “gamed.”

For example, many vendors release bundled security fixes (that is, a number of fixes are bundled into a single patch and released on a predictable schedule). There are a number of metrics a vendor can keep on these patches to help them answer some of the questions outlined about, principally, “Are we doing better or worse?” In fact, in the case of a security patch, there is no *single* metric that you could use to answer "goodness questions" related to how you are performing.

For example, one useful metrics would be "number and percentage of patches that were published on the announced date." Many customers plan their patching around announced dates so a vendor being able to “hit the date” allows customers to patch according to a published plan and with less disruption and cost. Also note that “time to fix a security issue” is not necessarily a good metric at all from a customer standpoint, because the metric that is relevant to actual *protection* – absent a workaround - is “how soon until customers actually apply the patch.” A “fix” that customers cannot apply (because of poor quality) or will not apply (because there are too many untested single issue patches that they cannot keep up with) is in some cases less useful than a patch that can be readily applied within someone’s planned maintenance cycle.

So, “number and percentage of patches published on announced date” is a useful metric for vendors to keep to see if they are meeting customers’ planning cycles and expectations. However, you do not want developers only hitting the date and ignoring quality. So, "number and percentage of patch *reloads*" is another metric that is useful, because while both vendors and customers want their “announced security patch bundles” to come out on the target date, vendors should not have to reload patches because the quality was poor (and customers do not want to have to reapply a faulty patch). Both quality and timeliness are important; hence, two metrics.

A tricky metric to draw conclusions about might be the number of issues contained in the patch. On the one hand, having a single patch that addresses a number of critical security issues is a potentially lower cost patching vehicle for customers. On the other hand, the larger the patch, the more testing everyone has to do. And the more issues you cram into a patch, the greater the chances something will break. What might be a useful metric to add to the combination is tracking how many issues drop out as the patch goes through testing. Ideally, once something has been identified as a Must Fix, you want to be able to deliver the patch with the issue addressed. So, over time, you want to get better at finding

the sweet spot of “fixing a number of critical issues, and having fewer drop out going through the testing process.”

In short, looking at a measurement area with a sense of “what are we trying to optimize,” considering how the metrics can be used as well as abused, and measuring a number of things to give you a better sense of your overall position, is better than fixating on a single number that will be gamed, especially if it is published and especially if vendors start competing on those numbers but are not required to disclose “the rest of the iceberg.”

The real goal of any metric is to *manage better*. Metrics can help you allocate resources to affect the most good for the most people and to spot trends (both positive and negative) quickly. Ultimately, a good metric needs to help you answer the question, "Are we doing better or worse?" You can do a lot with metrics, and some people lie with them, but above all, anyone using them has to be brutally honest with themselves. As Shakespeare put it so well:

This above all: to thine own self be true,
And it must follow, as the night the day,
Thou canst not then be false to any man.

Economic Value of Metrics

Cost effectiveness of metrics from those who are keeping them is a critical consideration, because there are already too many laundry lists of 5,000 things that everyone should do to be more secure. Even if all 5,000 of those things are somewhat useful, just like New Year's Resolutions, they are likely to be more actionable and "accomplishable" if the list is shorter. Putting it differently, nobody can do everything all at once, so for any organization, doing what is most achievable and valuable now and doing more later is a better way to make actual progress. Any 5,000 item initiative will fail due to the expectations - and failure to prioritize - that it entails.

It is a basic tenet of optimization problems that you can't optimize on all parameters. Time is constrained. Resources are (ultimately) constrained. Answering the question, "How can do X while making best use of scarce resources?" means I need to take account of what I most want to accomplish and how valuable is it to me that I accomplish those things.

As this relates to security initiatives around "what metrics and artifacts at every stage of development you should produce to 'prove' assurance claims," many people measuring the assurance of software believe that there are things you ought to be able to produce and measure at each stage of development. However, there is a cost to producing metrics and artifacts. If the cost of producing these is greater than the value of more information, no vendor can be expected to or should put the work in to produce them. Even if everything has some value, some things are more critical than others or provide greater

value for the work you put into getting them. One of the ways to tranche a metrics project is to look at

a) what can be data mined today to provide security metrics?
b) what else would we like to know (in some priority order)?
c) what will it cost to get that information? and
d) is the cost less than or greater than the benefit of the information?

If you are a small company, maybe you can't - in the beginning - do every single Best Practice Recommendation (or produce every single metric or every single artifact that anybody in a theoretically perfect world would want). But you can do something, and you'd be willing to do something if someone helped you by telling you what the most important things are to do first that make the biggest impact. Something is almost always better than nothing.

Those who want metrics or artifacts that are not economically viable for the vendor to produce should expect to pay more for the product or service, because if it is not actually helping the vendor manage better, it is not reasonable to expect them to produce something with no economic value to them. And asking for something that is not economically viable should be weighed against what is economically viable (and useful) in some priority order. Knowing how big the head of a pin is might be far more important than knowing how many angels can fit on it, and more readily obtained.