Labeling Software to Dramatically Reduce Vulnerabilities

By John Diamant, CISSP, CSSLP

founder of the secure development program for the largest technology company at the time (HP, prior to its numerous divestitures and splits), acknowledged by name for “significant comments and suggestions, which greatly improved” NISTIR 8151, “Dramatically Reducing Software Vulnerabilities: Report to the White House Office of Science and Technology Policy”, briefed senior staff of a Joint Congressional Subcommittee on software security, presenter at numerous conferences, and co-inventor on 11 issued patents (most security-related).

Position

The pervasive software (critical, IoT, and general consumer) vulnerability problem is *not* fundamentally a technical or logistical problem. It is first an economic problem, and can only be solved by fixing the economic incentives, which only *correctly implemented* labeling can accomplish, to change software vendor behavior.

I start with above because we first have to identify the correct problem that labeling needs to solve, in order to avoid a solution to the wrong problem which fails to address the fundamental issue. If labeling is viewed as a solution to technical problems rather than an economic one, the solution won’t move the needle on pervasively insecure software.

By above, I don’t mean the technical or logistical problems are easy or that they don’t need solving. Rather, I mean that the fundamental reason most software is vulnerable is broken economic incentives, for which labeling can be the solution, but only if done in ways which move the economic needle of purchase decisions.

In NISTIR 8151, “Dramatically Reducing Software Vulnerabilities: Report to the White House Office of Science and Technology Policy”, text reflecting some of my feedback is the highlighted part:

> “*While the user community clearly wants higher quality software, it is difficult for them to meaningfully ask for it and know if they received it, and thus signal the development of low vulnerability software. The market needs improved measures that are customer-focused, as well as other policy and economic approaches. For a measure to significantly inform customers, it requires pervasiveness, understandability, simplicity and efficiency. An example is the 5-Star Safety Rating of the National Highway Traffic Safety Administration (NHTSA). Once ratings consistently appeared on new automobiles, one- and two-star rated cars rapidly became scarce [Rice08].*”

Consumers and government purchasers (even experts) can’t tell how vulnerable the software (product embedded or otherwise) they purchase is, so they don’t select for low vulnerability software. As a result, software vendors are not incented to produce low vulnerability software, since it doesn’t help sales and may be perceived as an additional cost or cause of delay. If labeling changes so purchasers generally have information about the likely level of vulnerability of software, that will put powerful economic incentives in place for vendors to technically secure their software. Without it, the market will continue to reward vulnerable software, and it will remain pervasive.

The rest of this paper applies the above to the software labeling problem. In particular, security quality signals (referring to the economic concept of a signal), or labeling, **only if pervasive, understandable, simple, and efficient**, is a **necessary precondition** to incenting substantive work by vendors to dramatically reduce software vulnerabilities that can truly move the needle on secure software in the same way that understandable and simple 5 Star Safety crash test ratings required on the window sticker on new car sales dramatically improved automotive safety in only a few years.

Understandable and simple for consumers means modeling after 5 star ratings systems consumers are familiar with (Consumer Reports, Amazon, TripAdvisor, and NHTSA crash test ratings, for example). Security testing and assessment is much more complicated and nuanced, so it has to be mapped into something easily understood and simple to function as an **effective economic signal**, an example of which I propose below.

Pervasive means it has to be required like nutritional labels or surgeon general warnings, and thus visible to purchasers/consumers at the point of sale/packaging. By this, I do not mean that robust security testing must be required for every piece of software, but that transparent and simple disclosure of lack of such testing, and thus presumed vulnerability must be disclosed as part of the pervasive rating system if the tests and assessments themselves aren't performed. I realize that NIST alone cannot mandate labeling on consumer or IoT products, but it can create the rating system which could later be mandated for pervasive use. Other agencies (such as FTC under false or misleading advertising provisions of the FTC Act, or Congress) could mandate as appropriate once the labeling system is tested via voluntary use, and the labeling system might be mandated under FAR updates (say for critical software) already contemplated as a result of the EO.

Proposed sample of labeling implementation of position

My recommendation for the model on how to map security testing and assessment for assurance to the simple 5 star rating model is similar to how crash test ratings are done. Below is an *example*:

Overall score (e.g same for NHTSA crash tests): 1-5 stars, aggregated from individual scores (I recommend using the worst score from individual ratings, ignoring any N/As, since security is a weakest link problem).

Individual scores would be based on specific practices from the Secure Software Development Framework (SSDF), especially all the PW (Produce Well Secured Software) practices.

- Security Requirements Gap Analysis (PW.2, crash test analogue: driver frontal crash): 1-5 star
- Architectural threat analysis (PW.1.1, crash test analogue: passenger frontal crash): 1-5 star
- Code static vulnerability scanning or manual code review (PW.7, crash test analogue: side crash): 1-5 star
- Penetration and/or other dynamic application security testing (PW.8, crash test analogue: rollover): 1-5 star

Individual scoring for each above bullet (e.g. threat analysis, code review, etc) would be mapped as follows:

- Not tested:
  - If independently validated to be Not Applicable (e.g. no source code so no code scan): **N/A**
  - Otherwise: **1 star**
    - If the test is applicable but not performed, past history shows that undiscovered vulnerabilities are almost certainly present, so 1 star is a good proxy until it is tested
- Fully tested with all high severity issues fixed (or determined not exploitable or N/A by an independent expert): **3 star**
- Fully tested with all moderate and high severity issues fixed (or determined not exploitable or N/A by an independent expert): **4 star**
- Fully tested, all issues fixed (or determined not exploitable or N/A by an independent expert): **5 star**

A few notes on above example:

A) Scores could be lowered for partial testing (e.g. partial coverage, such as <100% of source code for static, less than 100% of code executed for dynamic, less than 100% of architecture include in threat model).

B) All relevant PW practices from SSDF are **required** for the same reason occupancy permit aren't issued with a passed electrical inspection but no roof or foundation engineering inspection. For software, the justification is even higher, as by now, most building inspections pass, whereas most software security assessments and tests fail (find moderately high severity and/or high severity issues on first assessment/test run).

C) Independent expert or testing could mean an objective 3rd party, which could increase score vs in-house expert.

D) I'll especially emphasize the importance of threat modeling (PW.1.1) as it is not performed nearly often enough in software development, and yet was raised numerous times throughout the critical software panels, and is understood by many experts, myself included, as a critical activity – see section 4.3.7 Threat Analysis in NISTIR 8151, a section I contributed.

Labeling to incent enterprise behaviors across numerous industry segments has been a defining characteristic of multiple security quality programs I've created and led. Labeling dramatically improved enterprise security quality, so I'm very familiar with what is effective, as well as how to develop, refine, and introduce labeling.