



Four Pillars of Software Security

Really: Four vendor practices that reduce risk from vulnerabilities

Copyright 2021 Carnegie Mellon University.

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

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon®, CERT® and CERT Coordination Center® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM21-0473

How does one measure software security?

Some incomplete answers

- This is hard to do
- Many practices, processes, technologies
- Different ecosystems, software systems, context
- Practically and economically unlikely to produce vulnerability-free software

How about “reduce risk from vulnerabilities” instead?

Observe vendor practices about

- Response capability
- Transparency

I. CVD

Coordinated Vulnerability Disclosure

Have at least basic Coordinated Vulnerability Disclosure capability

- Vulnerability Disclosure Program (VDP)
- Bug bounty
- Intake alone won't cut it
 - Look for vulnerabilities yourself first
 - Sort out internal capability first
 - Then consider external VDP or bug bounty

II. SOTA

Secure Over-the-Air Updates

Really: Secure updates

- Radio, copper, fiber, USB flash drives
- Automatic
 - Depending on ecosystem
 - Very important for consumer/commodity software
- Centralized risk
 - Note recent (and past) “supply chain” incidents
 - Security benefit probably outweighs the risk?

III. EOL

End of Life

Really: Period during which vendor will provide security updates and support

- Inform users and potential users
- Before, at time of acquisition
- With sufficient notice

IV. SBOM

More on this tomorrow!

<https://www.ntia.gov/sbom>

You're already doing it

- Upstream copyleft/GPL
- To build software, tools know what goes in

The ability to produce *any* sort of SBOM indicates vendor knowledge of upstream supply chain

More, older software implies more vulnerabilities

Why?

Hypotheses: If practice followed, risk is reduced

Users/customers might look for it

- Or request it
- Or require it

Yes please also do other “to the left” SDL things

- Threat modeling
- Architecture
- Language choice
- Fuzz/negative testing

Review

- I. CVD
- II. SOTA
- III. EOL
- IV. SBOM