A Brief Introduction to Developing Secure Software

David A. Wheeler
Director of Open Source Software Supply Chain Security, The Linux Foundation

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

- Twiddling configuration knobs isn't enough
- Most developers have *never* been taught how to develop secure software
  - "Software is more secure than we deserve"
- Brief summary of "what you should have been told" (& link to free course)
  - Basics, design, reuse, implementation, verification, & other
- Explain the OpenSSF & note some free OpenSSF materials to help you

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

What should you have been told?

- Security Basics
- Design
- Reusing Existing Software
- Implementation
  - Input, processing, calling other programs, output
- Verification
- Other: Cryptography, Vulnerability Disclosures

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Security Basics

- Security = Confidentiality, Integrity, & Availability
- Risk Management: Nothing is risk-free. *Manage* risks
  - Risk = likelihood + impact
- Need Protect (Identify & Protect), Detect, *and* Respond (Respond & Recover)

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Design

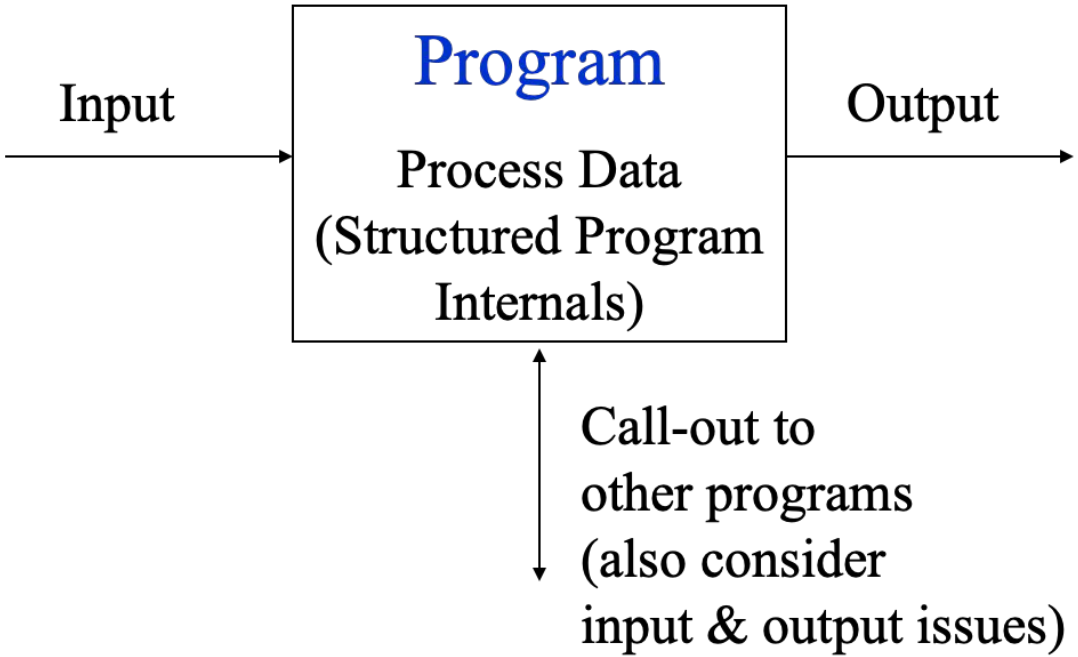
- (Architectural) Design = a program's top-level structure
- Least privilege: Each user/program should have fewest privileges possible
- Open design: Don't depend on attacker ignorance
- Non-bypassable: Security checks must *not* be bypassable (server-side check)
- Easy to use: If it's hard to use, users will bypass the security mechanism
- Harden system: Reduce likelihood that a single mistake is a vulnerability

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Reusing Existing Software

- Are you evaluating the intended version?
  - Counter typosquatting & dependency confusion attacks
- Is it maintained?
- Is there evidence that its developers work to make it secure?
- Is it easy to use securely?
- See more in “Concise Guide for Evaluating Open Source Software”
  - best.openssf.org/Concise-Guide-for-Evaluating-Open-Source-Software

Implementation



- Most vulnerabilities are common mistakes
- Learn what they are & how to avoid them
- Two helpful lists:
  - OWASP Top 10 (for web apps)
  - CWE Top 25

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Implementation: Input

- Identify *all* inputs from potentially untrusted users
- Validate *all* those inputs using an “allowlist” pattern, *not* a denylist
  - Give a picky pattern of what’s allowed; forbid everything else
  - A denylist requires you to predict all possible attacks - bad idea!
  - Regular expressions are often useful for this; match whole string (^...$)
- Ensure input checks are non-bypassable

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Implementation: Process data

- Avoid Default & Hardcoded Credentials
- Avoid Incorrect (Type) Conversion or Cast
- Memory safety is critical, use a memory-safe language if practical
  - In many organizations ~70% vulnerabilities from memory unsafe practices
- If you must use a memory-unsafe language (C/C++), take steps to reduce risk
  - Code to counter buffer overflow, use-after-free, double-free, etc.

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Implementation: Call out to other programs

- Try to avoid eval(), exec(), execute(), system(), etc. - easily misused
- Create SQL commands with parameterized statements - *not* concatenation
  - Fine special cases: prepared statements & correctly-implemented ORMs
  - "select * from authors where lastname = ?" … note the “?” parameter
- OS shell injection: Avoid calling the shell directly when it’s not necessary
- Use only documented APIs & check return results
- Log security-relevant events (login, logout, etc.)

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Implementation: Output

- In web applications:
  - Use frameworks with template engines that escape by default to counter Cross-site scripting (XSS) - if it's not the default, it's too hard to get right
  - Content Security Policy (CSP) - forbid inline JavaScript & CSS in HTML
  - Use HTTP hardening headers
- Don't let attackers control format string parameters, e.g., printf(PARAM, ...)

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Verification

- Use static analysis tools to examine source code for problems
- Use dynamic analysis tools (e.g., fuzzers & web application scanners)
- Use Software Composition Analysis (SCA) tools - detect dependencies with known vulnerabilities
- Have a good test suite so you can update with confidence
  - Include “negative tests” - ensure that what shouldn’t work, doesn’t work

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Other: Cryptography, Vulnerability Disclosures

- Cryptography
  - Never develop your own cryptographic algorithm or protocol
  - Never implement your cryptographic algorithms or protocols
  - Ensure what you choose is strong & configured correctly (not ECB mode!)
- Vulnerability Disclosures
  - Make it *clear* how to report vulnerabilities & be ready for them

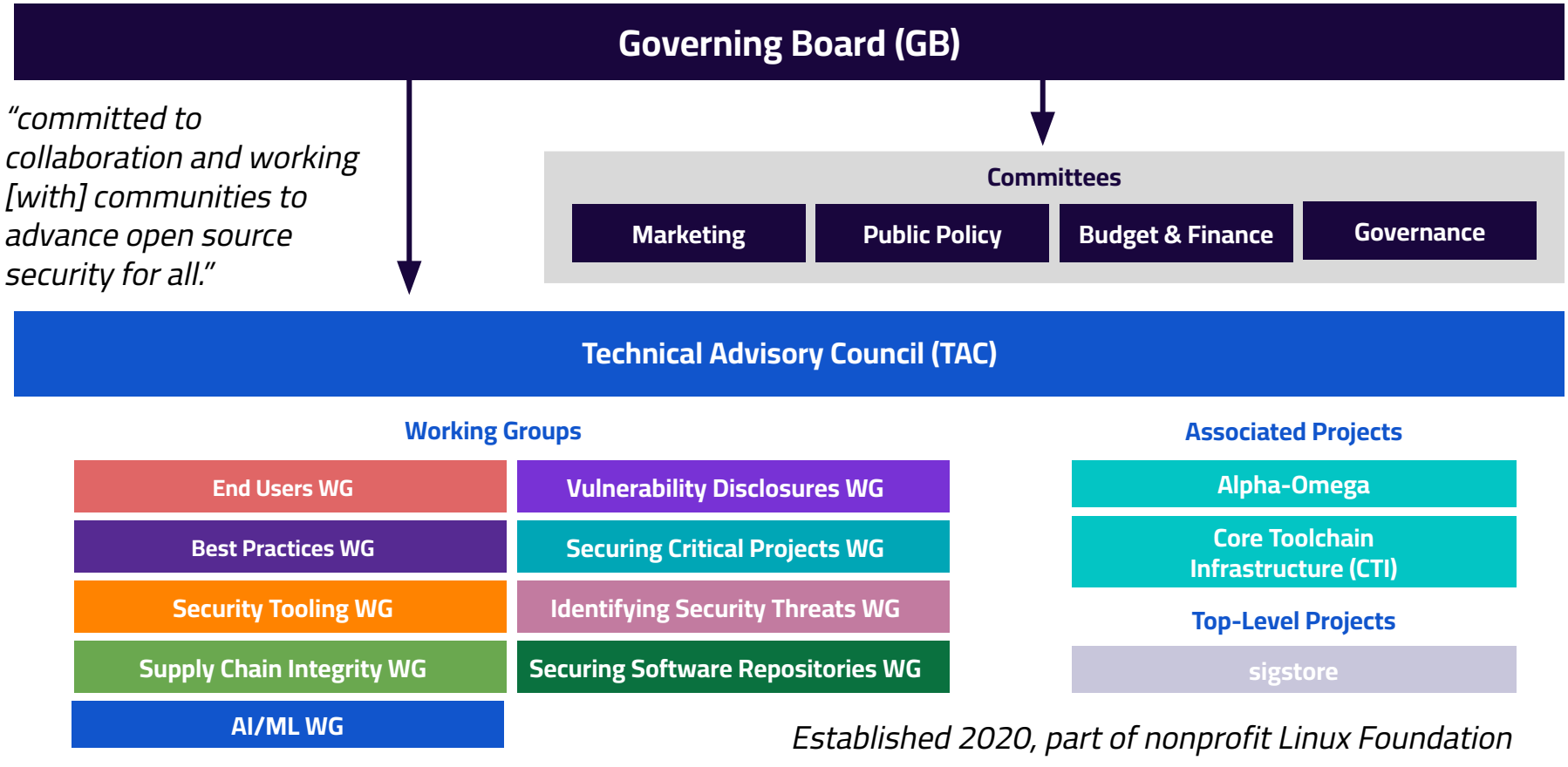
OpenSSF
OPEN SOURCE SECURITY FOUNDATION

To learn more, take our free course!

"Secure Software Development Fundamentals"
https://openssf.org/training/courses/

Free & can earn certificate of completion

Open Source Security Foundation (OpenSSF) Structure



Established 2020, part of nonprofit Linux Foundation

Sample OpenSSF Results

- Education: *Secure Software Development Fundamentals* (**free** course)
- Guides:
  - *Concise Guide for Developing More Secure Software*
  - *Concise Guide for Evaluating Open Source Software*
- OSS Security Evaluation:
  - *OpenSSF Scorecard*; auto-measures OSS github.com/ossf/scorecard
  - *OpenSSF Best Practices Badge* (for OSS projects); >6,100 participating, 3 levels
  - *Supply-chain Levels for Software Artifacts (SLSA)*
- Improved tooling: *Sigstore (signing)*
- Vulnerability finding/reporting:
  - *Alpha-Omega:* proactively find/fix vulnerabilities openssf.org/community/alpha-omega
  - *Vulnerability Disclosure Guide* github.com/ossf/oss-vulnerability-guide

OpenSSF Scorecard

- Automatically scores OSS projects on heuristics ("checks")
  - Each related to security, scored 0-10, weighted average computed
  - Can use to evaluate your own or others' projects (they don't need to cooperate)
  - Works projects hosted on GitHub & more recently GitLab
- We routinely run Scorecard on > 1M OSS projects; any can run
- Sample checks (out of 19):
  - Binary-Artifacts — Is the project free of checked-in binaries?
  - Branch-Protection — Does it use Branch Protection?
  - CI-Tests — Does it run tests in CI, e.g. GitHub Actions, Prow?
  - Code-Review — Does it require code review before code is merged?
  - Contributors — Does it have contributors from at least two different organizations?
  - CII-Best-Practices — Does it have an OpenSSF (formerly CII) Best Practices Badge? [next!]
- Sonatype 2022 report found it could help predict likelihood of known vulnerabilities
- https://github.com/ossf/scorecard

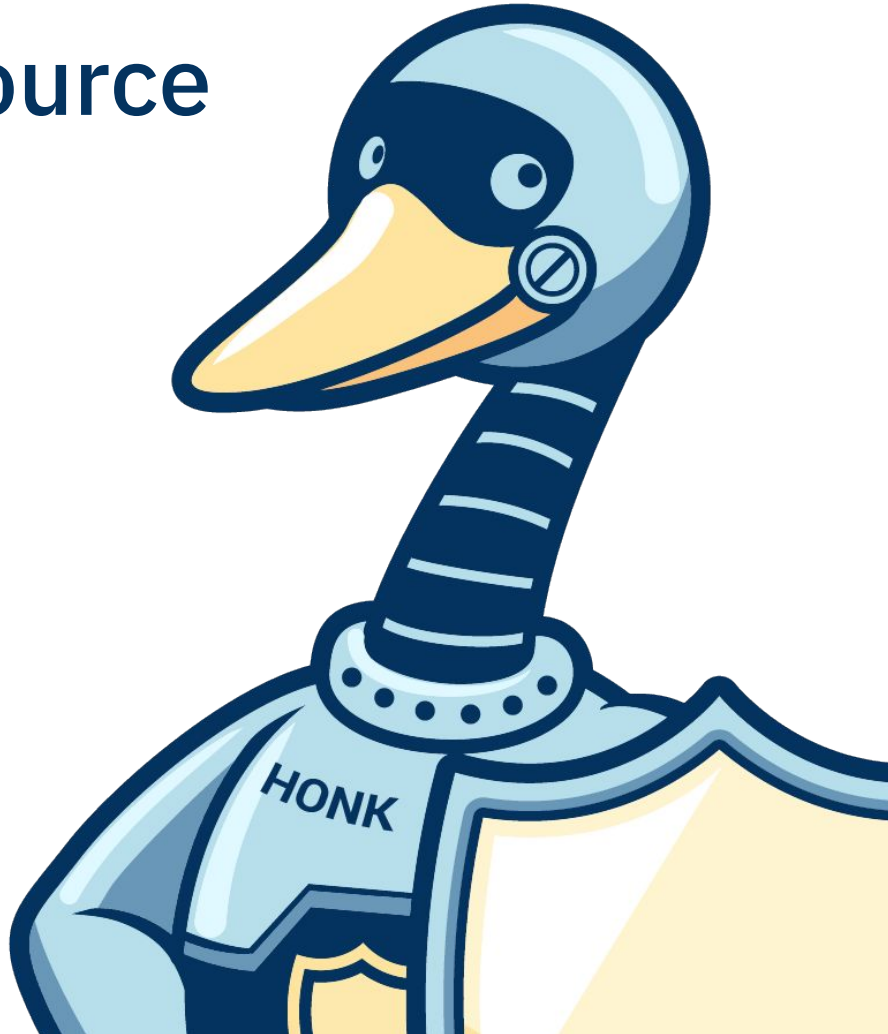
OpenSSF Best Practices Badge

- Identifies best practices for OSS projects
  - Goal: Increase likelihood of better quality & security. E.g.:
    - "The project sites… MUST support HTTPS using TLS."
    - "The project MUST use at least one automated test suite..."
    - "At least one static code analysis tool MUST be applied…"
    - "The project MUST publish the process for reporting vulnerabilities on the project site."
- Form-based approach based on practices of well-run OSS projects, with some automation
- If OSS project meets best practice criteria, it earns a badge
  - Enables projects & potential users know current status & where it can improve
  - Combination of self-certification, automated checks, spot checks, public accountability
- Three badge levels: **passing**, **silver**, **gold**
- Participation widespread & continuing to grow
  - >6,100 participating projects, >1,000 passing+ (Sep 2023)
  - Current statistics: https://www.bestpractices.dev/en/project_stats
- For more, see: https://www.bestpractices.dev

Interested in Open Source Software Security?

Please get involved in the Open Source Security Foundation (OpenSSF) - openssf.org

- Join a Working Group / SIG / Project
  - Slack channels
  - Mailing lists
  - Online meetings
- Have your organization join as a member!



OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Get our latest news

https://openssf.org/sign-up

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Follow us on Social Media

[Twitter](#)
@openssf

[LinkedIn](#)
OpenSSF

[Mastodon](#)
social.lfx.dev/
@openssf

[YouTube](#)
OpenSSF

[Facebook](#)
OpenSSF

Thank You

This presentation released under the CC-BY-4.0 license