Attestations style guide

Welcome to the style guide for displaying attestations on package repositories.

This guide presents the findings from our user research, including summaries, clarifications, and rationale. Based on this, we provide 'good first' UI examples and recommendations to guide initial adoption by three software package registries: PyPI, RubyGems.org, and npm. We're highlighting these registeries because they are among the most widely used and actively evolving open source package ecosystems, each with distinct approaches, and each serving a wide range of user and project types. Their prominence within the broader software supply chain makes them representative models for attestation UX and documentation.

This project was initiated by the OpenSSF Securing Software Repositories working group to improve the user experience (UX) of attestations. For more context, you can review the initial request and the project task board:

- **Initial Request:** https://github.com/ossf/tac/issues/424
- **Project Task Board:** https://github.com/orgs/ossf/projects/36

While we focused on only the three registries for actionable detail, the guidance and templates here are designed to be adaptable. So we encourage maintainers and contributors from other ecosystems to apply and extend these principles for fostering secure and consistent repository practices to their chosen registeries. In fact, these recommendations are intended to be flexible enough to also work outside of package registry page layouts and should be iterated on and improved. For information on how to extend this work, please see the contribution page of this guide.

Work plan & team

The public workplan can be seen here: https://github.com/orgs/ossf/projects/36?pane=issue&itemId=115907793&issue=ossf%7Cwg-securing-software-repos%7C72

The individuals /organisations involved in this project are listed below.

- Open SSF
- OpenSSF Securing Software Repositories working group
- Kabu Creative
- Superbloom Design
- Implementation partners TBC
- Our 15+ User Testers from the OSS community

Contents

This style guide contains the following sections:

1. How to use
2. Lowest UI requirements (Level A)
3. Medium UI requirements (Level AA)
4. Highest UI requirements (Level AAA)
5. Attestations, Build, Source, Integrity UI in detail
6. Additional signals of trust
7. Icons
8. Hyperlinks and linking information
9. Documentation
10. Language localisation
11. Contribution information

Research

We conducted three phases of research with users and additionally secondary (desk) research throughout the project.

Phase 1 The first phase was exploratory research focused on how users find security information on packages and their initial perceptions of attestations. Our methods were based on user interviews, behavioral observation, and feedback on existing, live content.

Phase 2 & 3 The second and third phases involved usability testing the initial UI designs for attestations. We also gathered feedback on the accessibility and clarity of the proposed documentation.

Research documentation: https://github.com/orgs/ossf/projects/36/views/3

More on secondary research: https://www.nngroup.com/articles/secondary-research-in-ux

Methodology

With the exploratory research we followed **contexual inquiry** and **semi-structured interview** methods. https://www.nngroup.com/articles/user-interviews/ & https://www.nngroup.com/articles/contextual-inquiry/

To synthesize our findings from the exploratory research we used **Thematic analysis** and **Affinity Diagraming** https://www.nngroup.com/articles/affinity-diagram/

Finally, we used **usability testing** and **preference analysis** to evaluate the proposed designs and documentation (see https://www.nngroup.com/articles/usability-testing-101/ and https://www.youtube.com/watch?v=-XYTk5MVvK0).

Attestations personas

To guide our research, we developed three personas to represent the types of users who might interact with attestations on package repositories. Each persona includes information about the user's role, goals, frustrations, and desired project outcomes.

1. **The Security Architect (concerned / informed)**
Represents a security expert who is highly informed about attestations and has deep foundational knowledge of software security. This user's primary goal is securing the software supply chain as a key part of their role.

2. **The Pragmatic Developer (somewhat concerned and somewhat informed)**
Represents a user who is aware of attestations but must balance security against many competing priorities. They may be an engineering team lead, senior software engineer, or DevOps engineer responsible for building and shipping products.

3. **The Incidental Consumer (not concerned, not informed)**
Represents a user who is unaware of attestations, because for them, code is a tool, not their profession. They may be a data analyst using Python, a designer using a JavaScript static site generator, a hobbyist, or a student following a tutorial.

View the full personas here:
https://github.com/ossf/wg-securing-software-repos/issues/70

More on personas:
https://www.nngroup.com/articles/personas-study-guide/

Research Results

You can find raw, unedited notes from our user research here:

https://github.com/ossf/wg-securing-software-repos/issues/66

https://github.com/ossf/wg-securing-software-repos/issues/83

The insights from this research informed the recommendations in this guide, including findings about:

1. User interface preferences
2. How users assess package safety
3. Comprehension of technical terms and concepts
4. Where and why users seek information
5. Opinions on attestations, their impact on trust and what other information, nearby attestations, help the trust of attestations.

How to use the style guide

The following pages in this style guide provide recommendations for displaying User Interface (UI) elements for attestations. We also identify and share additional elements that users found useful for building trust in the safety, security, and integrity of the packages they install.

All recommendations are grounded in user research and testing, conducted with users who reflect each of the personas detailed in the style guide introduction.

We understand that each package registry has unique structural and design considerations. Therefore, the extent to which these guidelines are implemented is at the discretion of each registry. We've scaled our UI recommendations to accommodate various section sizes and the distinct ways packages are presented. For example, some registries embed extensive READMEs directly on package pages, pushing down other content. Others feature wider main sections with smaller sidebars. Others include infrastructure for multiple pages or menus, making it easier to integrate new pages and sections.

We recommend starting with the implementation that best suits your package registry/ platform while maintaining the consistency of your existing visual design system. We advise against altering core branding elements such as colors, hyperlink styling, or border properties (size, color, radius). Our recommended UI elements are designed for flexibility, and should be integrated into most existing systems without requiring fundamental changes to your visual language. There is always the opportunity to contribute a new iteration along with your rationale.

In the spirit of open source, and where practical, all project materials - from research protocols and personas to anonymized user testing notes and UI designs in Penpot - are publicly available. Please refer to our contribution pages for more details.

6. Explain common security attacks and how they are mitigated

Transparently list the threats the user faces (e.g., Typosquatting, Account Takeover) and what they can do to protect themselves. Explain platform-level features that mitigate common attacks, including the role of Trusted Publishing and attestations.

Provide clear reporting channels for users concerned about the security of an individual package or the package repository as a whole.

7. Be transparent about limitations and set clear expectations

Documentation must be upfront about the scope and limitations of the security practices or features it describes.

State clearly that an attestation proves *build provenance* (where a file came from) but does not vouch for the *code's safety* (that the code is free of bugs or vulnerabilities). Explain that trusting an attestation means implicitly trusting the tooling and authority that issued it.

Be transparent about the readiness of downstream tooling. Where applicable, highlight that automated attestation verification is not yet ready in common package managers.

4. Warn package consumers that packages are not secure by default

Some package consumers hold the incorrect assumption that because a package is published on an official repository it is secure and/or trustworthy.
To combat this, include disclaimers on pages for package consumers, directing users towards security documentation. Example from PyPI:

> *Warning: Installing packages from PyPI means running third-party code, which carries inherent security risks. Before installing any package, we strongly recommend reading our Package security guide.*

5. Normalize secure-by-default publishing workflows

Documentation should be opinionated in favor of security. The most secure method for any task should be presented as the canonical, recommended path.

When writing documentation about publishing packages, prominently feature the most secure, automated publishing method that generates attestations. This should be the first and most detailed workflow described, establishing it as the modern standard.

Where applicable, clearly and explicitly state that using the recommended workflow confers the benefit of generating provenance attestations *automatically*. This makes adopting the secure workflow more compelling.

Legacy methods that do not produce attestations should be explicitly marked as "deprecated" or "for advanced use cases" to guide users away from them.

2. Structure documentation around user roles

Package consumers and maintainers require (and have) different levels of knowledge about security concepts and features, including attestations.

Content for package consumers should be separated from content for package maintainers, surfacing relevant and actionable information to each audience.

Structure the Information Architecture around user roles, using clear, top-level sections like "Using This Repository" (for consumers) and "Publishing Packages" (for maintainers).

3. Provide content to help package *consumers* understand attestations within the context of other security features

Attestations are only one part of a larger security story. To be effective, documentation must place them in context, helping users build a holistic mental model of risk.

Documentation for package consumers should explain how security practices and features work together as a layered defense. It should be structured around the fundamental questions a user has about risk:

1. Can I trust the source code?
Explaining signals like project activity, vulnerability scanning, etc.

2. Where did this package come from and how was it made?
Explaining build provenance and attestations

3. Did I download the same file that is hosted on the package repository?
Explaining checksums and integrity

Build Provenance
- Definition: A verifiable trail of evidence for a package's origin. It answers where a package came from and how it was built
- Do: Clarify this term in headings or text when the concept might be new to users
- Example: "Verifying build provenance (package origin)"

Build Provenance Attestation
- Definition: A specific type of attestation that describes how, when, and where a package was built
- Do: Clarify the relationship between attestations and build provenance using this term
- Example: "A build provenance attestation describes how, when, and where a package was built."

Transparency Log Entry
- Definition: The tamper-proof record of the build provenance attestation saved to a public ledger
- Do: Use "transparency log entry" to describe how an attestation is recorded
- Example: "Each attestation is recorded in a public ledger as transparency log entry, which is a permanent, auditable record that cannot be changed."

Verified
- Definition: The process that a package repository takes when accepting an attestation from a build platform, ensuring the attestation's source matches the maintainer's configuration
- Do: Use "verified" to describe the repository's automated check during package upload
- Example: "PyPI verifies this attestation at upload time, confirming that the identity matches what the maintainer previously configured for Trusted Publishing."

Best practices for documenting attestations

Attestations documentation should:

- Help package consumers understand attestations, including what they are, how they work, and what their limitations are

- Help package consumers understand attestations within the context of other security features and practices

- Direct package maintainers towards workflows that generate attestations

For attestation documentation to be most successful, we recommend documentation authors follow these seven principles:

1. Use key terms consistently

To ensure clarity and consistency for attestation consumers, use key terms as defined below. Adhering to these standards prevents user confusion across packaging ecosystems.

Attestation
- Definition: A statement of signed facts (or metadata) about a software artifact
- Do: Use this as the general term for a signed claim. It is the foundation for more specific terms like "Build Provenance Attestation"

Style guide: Contributing or iterating on this work

This is the first iteration of UI recommendations for attestations and supportive relevant information for all levels of developer persona understanding.

We encourage continued contribution to this work, including:

1. Continued user research that can expand on and/or follow the guidelines set out in previous User Research: https://github.com/ossf/wg-securing-software-repos/issues/82 and https://github.com/ossf/wg-securing-software-repos/issues/66

2. Developing alternate, refined and appropriate UI iterations that are tested with users for clarity and usefullness in terms of supporting their safety and security trust in packages: https://github.com/ossf/wg-securing-software-repos/issues/65

3. Improving and iteration on documentaton templates and examples: https://repos.openssf.org/

4. Enhancing the graphic design of unique icons and developing a specific attestation logo/symbol. (see the Icons page of this style guide).

5. Any other contributions and/or recommendations can be brought to the OpenSSF Working Group on Securing Software Repositories https://github.com/ossf/wg-securing-software-repos