Hello. My name is Dan Shernicoff and I am a security research engineer at Tenable. I spend my days looking at security advisories and teaching computers how to read them (no AI, just basic parsing). And today I'm going to give you some insights into what makes a good security advisory as well as some examples of what not to do.

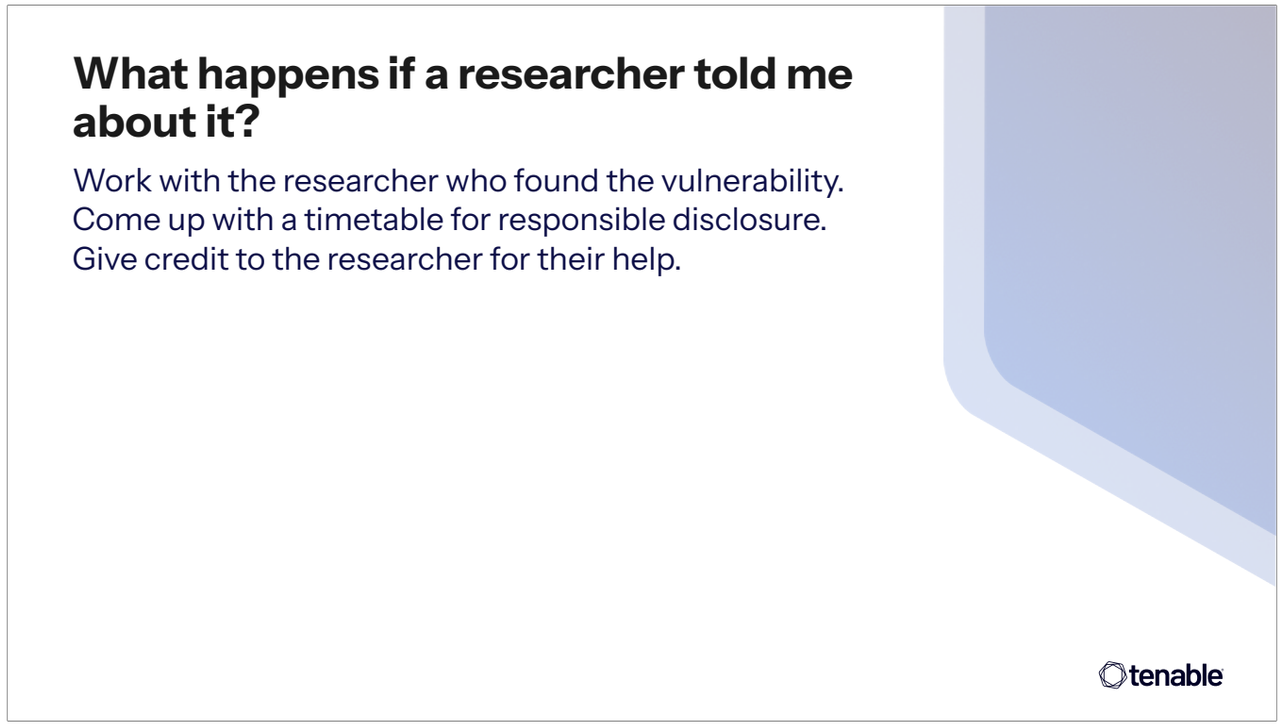
First things first

Fix it!!!

Get the fix out, as quickly as can be.

tenable

You found out you have a security vulnerability in your code. It doesn’t matter if you found out through internal testing, because a security researcher reached out to you, or if it’s a zero day that has been exploited before you discovered it. The first thing you need to do is fix it. Do not leave your users in a vulnerable state for any longer than they need to be.
Once you’ve done that you need to let your users know about it and what steps they need to take to address it.



When a security researcher finds a vulnerability they will reach out with what they have found.
Work with the researcher to make sure that you understand the issue and how to reproduce it.
Come up with a timetable for responsible disclosure, industry standard is usually within 90 days.
Don't forget to tell people who helped you find and fix it.
If the researcher asks for compensation, talk to your legal team about whether it's appropriate. If you don't have a legal team or can't afford to give cash, stickers are worth twice their weight in gold.

Now that you've fixed the vulnerability, or figured out how it can be mitigated until a fix is ready, you need to tell people. But who do you need to tell?

1. Your users. They need to know what to do to not be vulnerable any more. This could include applying an update, changing their configuration, or in the case of a library, what code changes they need to make.
2. SecOps. These folks work hard to make organizations' computers and networks secure. They want to know what can be done to remediate the issue, just like users, but they want more information so they can prioritize things properly. They want to know if physical access is needed. They want to know what exploiting the vulnerability allows. They want to know how severe it is and how likely it is to be exploited.
3. Automated systems. SecOps relies on tools that help them determine what vulnerabilities exist in their systems. They leverage systems that scan the networks and devices that they are in control of to see what vulnerable software is installed (I say "is installed" because I can guarantee that every computer, everywhere, has at least one piece of software with a vulnerability on it.) These systems rely on automation to ingest security advisories to be able to properly run the scans and report accurately.

When you're writing a security advisory you need to keep all of these groups in mind.

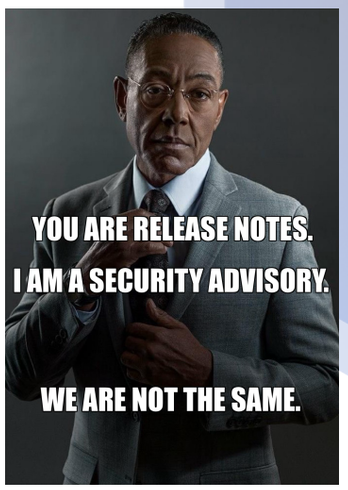
Security Advisories are not Release Notes

Release notes and security advisories serve different purposes.

Release notes are to let users know what changed in the release.

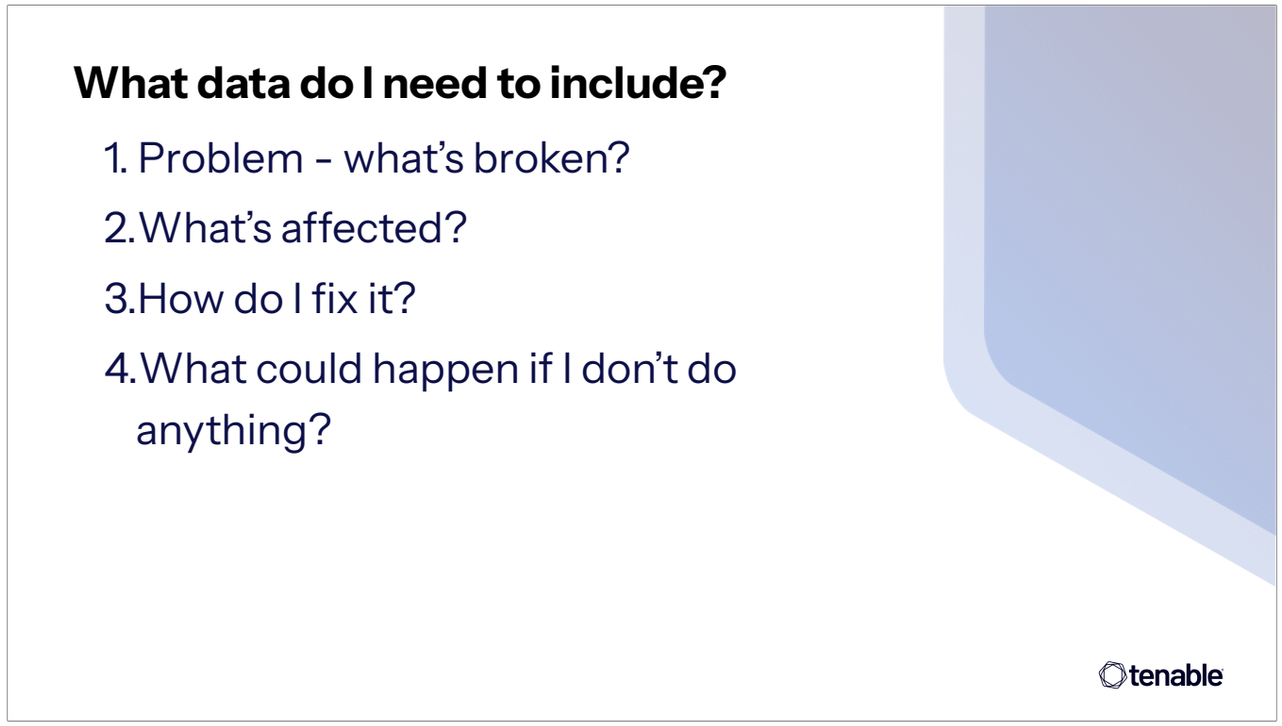
Security advisories serve to let people know what security issues were found and (hopefully) addressed.

Security advisories and release notes should reference one another but they are not interchangeable.



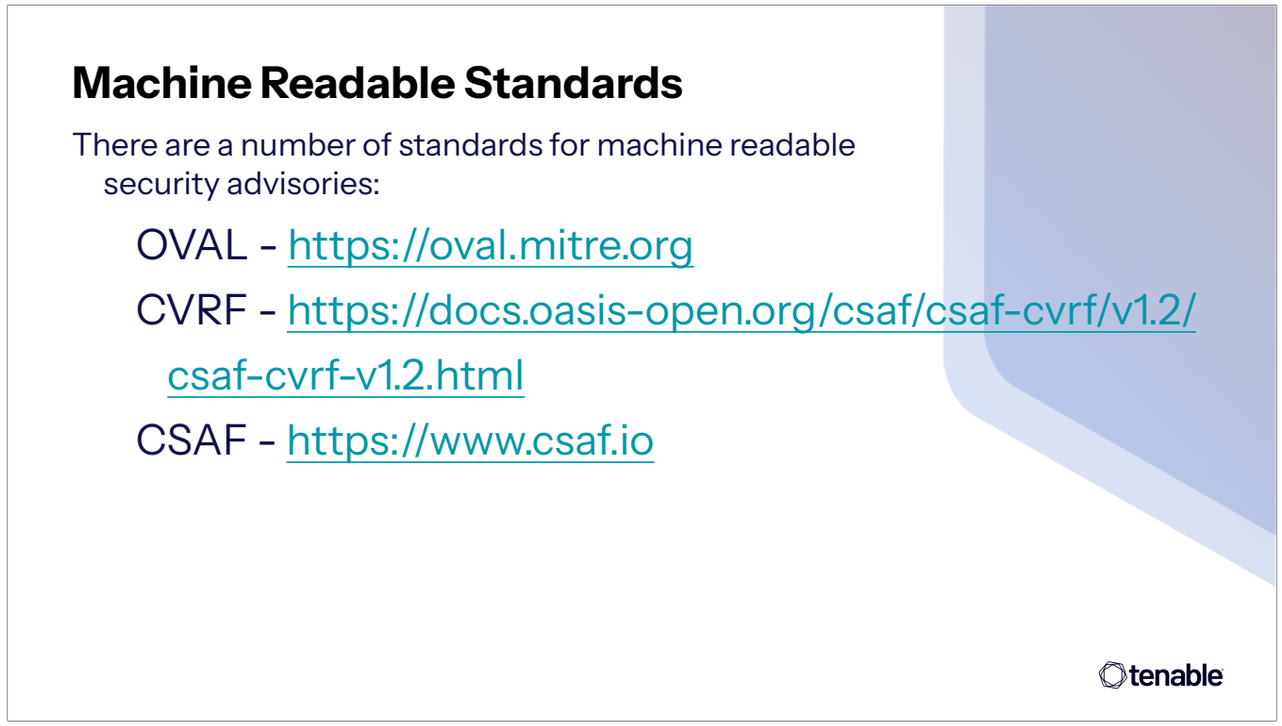
tenable

This should probably not need to be said but there is a difference between security advisories and release notes.
Release notes tell people about everything in the release - new features, bug fixes, and security fixes.
Security advisories are just about what the security issue is and how to address it.
Yes they should reference one another but you shouldn’t have one thing that serves both purposes.



Now let's talk for a minute about what needs to be in the advisory.

1. What is the problem? Can be something like "A malformed query can lead to remote code execution." It does not necessarily need to be overly detailed but it should give an indication of:
    a. Whether physical access is required.
    b. If it can cause arbitrary code execution.
    c. Will data leakage or corruption occur.
2. What's affected? Which version or versions have this issue. If a user is on version 2.3 and the vulnerability only came in in 2.4 then they are fine. If you're not sure, assume the vulnerability exists. Better to say "All 2.X versions" then to say "Version 2.4" if you are not 100% sure that's correct.
3. How do I fix it? Is there an upgrade? A configuration change? What can I do to no longer be vulnerable?
4. What's the impact of leaving this in place? This is where you might want some additional details to the questions addressed in part 1.



There are a number of standards that have been created for writing security advisories that are machine readable.

OVAL is an XML format, primarily used by linux vendors, for security advisories. It allows for a single XML file with multiple advisories embedded in it. Each advisory is encapsulated in a “definition” and will include pointers to tests, variables, and objects that might be used across multiple advisories.

CVRF is also an XML format, however each advisory is generally its own file. CVRF has been used by many organizations to provide their security advisories in a machine readable format.

CSAF, or CVRF version 2, is a JSON format that many organizations are moving towards. Similar to CVRF it is 1 file per advisory. Many organization that have used CVRF in the past, as well as some linux vendors, are transitioning to using CSAF for their advisories.

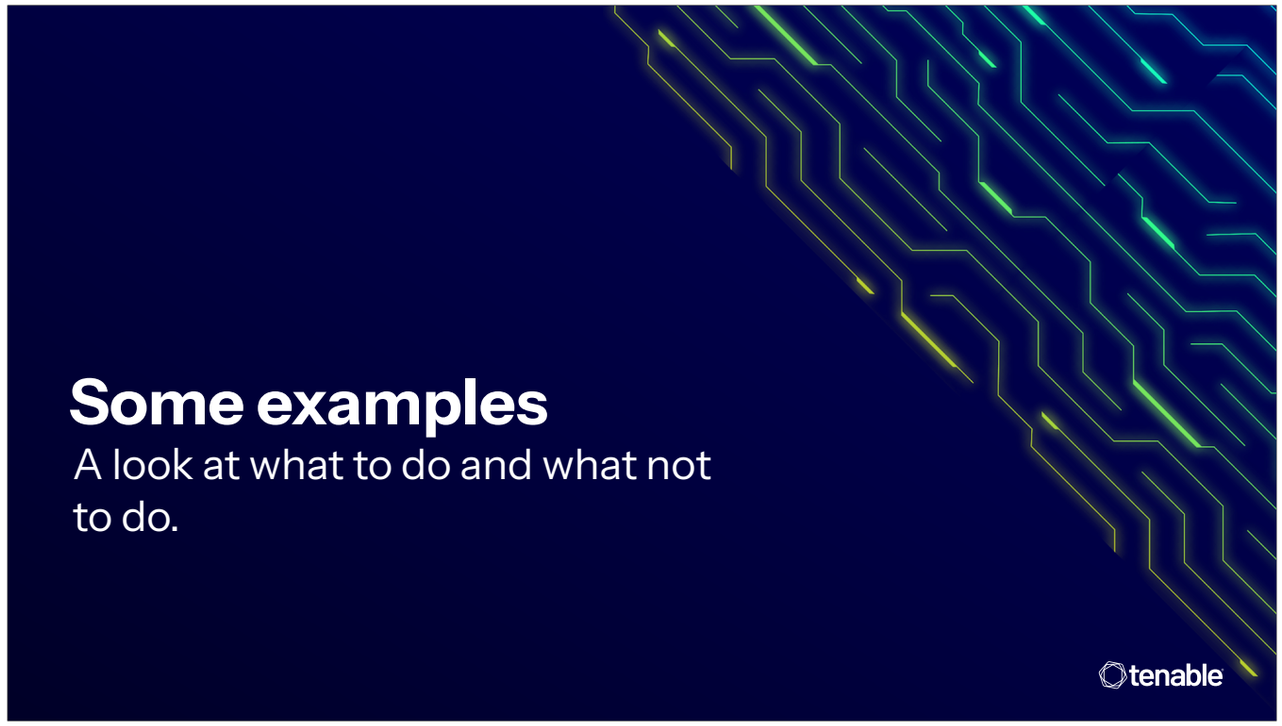
All of these standard formats provide all of the information that needs to be in an advisory.

Is machine readable (XML, JSON, etc.) really necessary?

It depends.

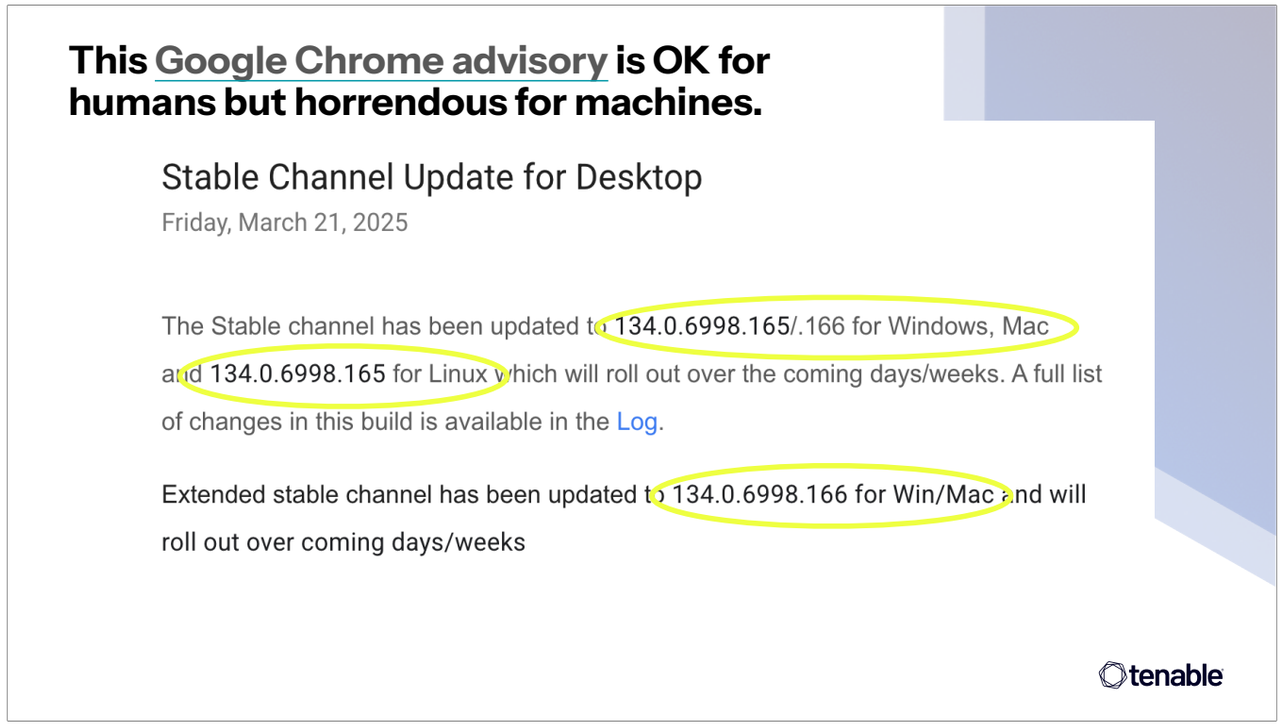
Regardless of whether it is or not, machine parsable is.

If you’re asking, “Do I need to write advisories in JSON, XML, or some other purely structured format that machines know how read?”
the answer is, “Probably not.”
Unless you’re writing them for an organization with the resources to devote to a PSIRT team, you probably can’t afford the overhead.
With that, you can make something machine parsable by using structured formats, like HTML or Markdown, and being very consistent in how it’s formatted. This includes things like CSS tags, element IDs, and document structure that computer systems can consistently key off of to know what data is where.
If your goal is something that both a human and a machine can parse, consistency helps with both.



Let’s take a look at some examples, first of what not to do, and then some that do it well.

This Google Chrome advisory is OK for humans but horrendous for machines.

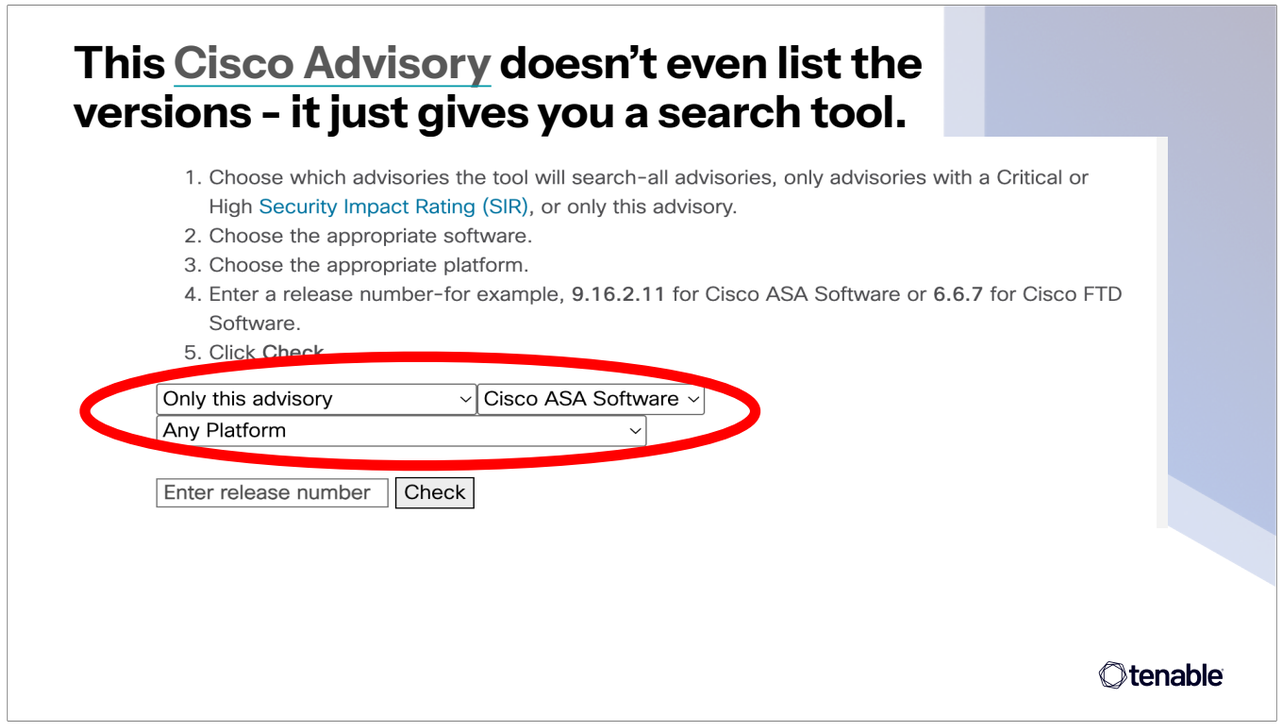


This formatting makes it hard to determine which versions are fixed for which operating systems.

Reading it I can understand that I should be OK on Windows and MacOS with either .165 or .166 but for Linux I need .165.

You can also note that for the “Stable Channel” it specifies “Windows, Mac” but for the “Extended Stable” it says “Win/Mac”.

These inconsistencies make it much harder to parse.



This advisory doesn't actually list the versions anywhere. It gives you a way to check to see if the version you are running is vulnerable.
For obvious reasons, a machine can't parse this. For SecOps people managing installations with lots of devices, probably multiple models, that are potentially affected and at different versions, it would take a long time to determine which of them are impacted.

Adobe uses well structured, consistent HTML for their advisories.

Affected Versions

Product	Track	Affected Versions	Platform
Acrobat DC	Continuous	25.001.20428 and earlier versions	Windows & macOS
Acrobat Reader DC	Continuous	25.001.20428 and earlier versions	Windows & macOS

Product	Track	Updated Versions	Platform	Priority Rating	Availability
Acrobat DC	Continuous	25.001.20432	Windows and macOS	3	Release Notes
Acrobat Reader DC	Continuous	25.001.20432	Windows and macOS	3	Release Notes

Vulnerability Details

Vulnerability Category	Vulnerability Impact	Severity	CVSS base score	CVSS vector	CVE Number
Use After Free (CWE-416)	Arbitrary code execution	Critical	7.8	CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H	CVE-2025-27174
Access of Uninitialized Pointer (CWE-824)	Arbitrary code execution	Critical	7.8	CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H	CVE-2025-27158

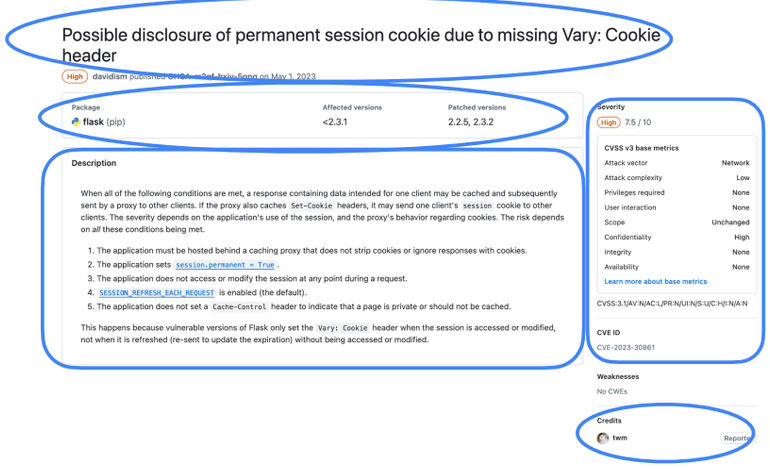
tenable

By using tables for their details Adobe makes it easy for both a human and a machine to pick out the relevant data such as:

- Which versions are affected
- Which versions are fixed
- What CVEs are affecting them and their individual severities

By breaking it all out it’s easy for someone to look and see if they’re affected and what to do to fix it.

This Flask advisory is just one example of how GitHub lets you see all the relevant details.



GitHub security advisories, which are available to all repos on GitHub, are consistently formatted with all of the fields that an advisory would need including
- Title
- Package and version information
- Detailed description
- CVE and severity data
- And acknowledgements

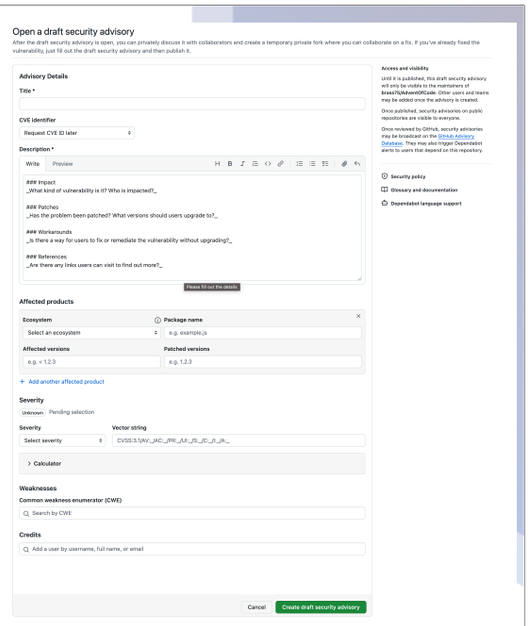
If you are using GitHub to manage your code you can leverage their security advisories. This will actually give you the best of both worlds
- A well formatted, HTML advisory that has all the data needed in easily identifiable sections
- The same data is available in JSON format via an API that machines can ingest.

You even get their GraphQL API if you're so inclined.

GitHub Security Advisories

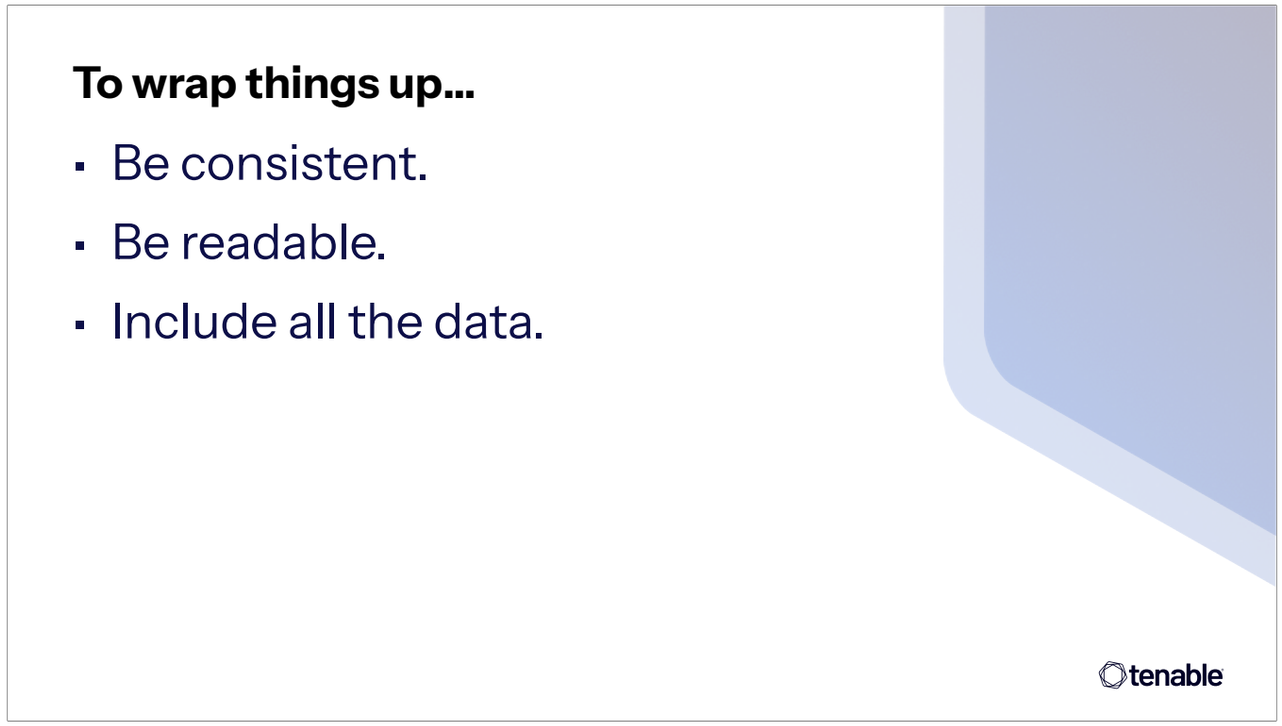
Filing a security advisory in GitHub is as simple as filling out a form and publishing it.

Once you have published the advisory GitHub will review it for use with Dependabot and inclusion in their advisory database.



tenable

Writing a GitHub security advisory is as simple as filling out a form. You can do this well ahead of publication as it will remain in a draft state until you choose to publish it. All of the details that we discussed earlier are on the form.
Once you have published, the advisory will be visible to anyone going to the "Security" tab of your repo. GitHub will review the advisory for inclusion in its Advisory Database and use in Dependabot.



To summarize:
Be consistent - in the phrasing and the formatting so that humans and computers can parse the data you are providing.
Be readable - Make sure that everything is intelligible. Try to avoid jargon. The people reading this may or may not know what you are talking about if you do.
Include all the data - what’s affected, how bad it is, and what to do to fix or mitigate it.

Dan Shernicoff

Reach out to me...
dshernicoff@tenable.com
https://twit.social/@brass75 - Mastodon
https://bsky.app/profile/brass75.brassnet.biz - BlueSky

Get the presentation deck...
https://github.com/brass75/PyCon25Talk/ - Repo with presentation
https://bnet.boo/pc25ms - This presentation (or just use the QR code over there 👉)

Download the slides

tenable

Feel free to reach out to me or download a copy of this deck. The QR code points to the Keynote file, the repo has a PDF version as well. Thank you very much.


tenable

CVE

The CVE (Common Vulnerabilities and Exposures) program is a program to help centralize information on cybersecurity vulnerabilities. It is currently run by MITRE and funded by the US government.

You can find more information on the program at https://www.cve.org

tenable

CNAs and getting a CVE

A CNA (CVE Numbering Authority) is authorized to assign CVE identification numbers. The PSF is a CNA for a number of projects. If your project is not in that list you have some other options:

GitHub if your repository is hosted there

Red Hat

For more information about the PSF as a CNA, please reach out to Seth Larson at cna@python.org

Python CNA information:
- https://www.python.org/cve-numbering-authority/

GitHub CNA information:
- https://docs.github.com/en/code-security/security-advisories/working-with-repository-security-advisories/about-repository-security-advisories#cve-identification-numbers

Red Hat CNA information:
- https://access.redhat.com/articles/red_hat_cve_program

tenable

CVSS

CVSS (Common Vulnerability Scoring System) is an industry standard way of showing what the severity of a given vulnerability is. It includes information on the vector, complexity, impact, etc. as well as a score from 0.0 - 10.0 which indicates, at a glance, how severe a given vulnerability is.

CVSS v3.1 Specification:
- https://www.first.org/cvss/v3-1/specification-document
CVSS v4.0 Specification:
- https://www.first.org/cvss/v4-0/specification-document

tenable

The PSF and Security

The PSF Security Developers in Residence, Seth Larson and Mike Fielder, work to advance security in everything Python.

PyPA is the Python Packaging Advisory Database which lets you file an advisory via a simple YAML file.
https://github.com/pypa/advisory-database

OSV - The Open Source Vulnerability Database - is a database of open source vulnerabilities that includes vulnerabilities published via GitHub
https://osv.dev/vulnerability/GHSA-3vcg-j39x-cwfm

via OpenSSF Malicious Packages
https://osv.dev/vulnerability/MAL-2025-3863

and Via PyPA
https://osv.dev/vulnerability/PYSEC-2025-39

tenable