OSS Risk Management

Petr Špaček, Victoria Risk – NANOG 91, June 11, 2024

Internet Systems Consortium, Inc.

Open source developer of BIND 9, ISC DHCP and Kea DHCP

(*NOT ISC2*)

https://www.isc.org

Government experts are here to help ...

- NIST Secure Software Development Framework (SSDF)
- EO14028 – Securing Critical Software
- NTIA Software BOM requirements
- CISA Secure by Design Pledge
- EU Cyber Resilience Act
- White House EO on Zero Trust (encryption for DNS and HTTP)

NANOG™

How do YOU assess OSS quality?

- We created a survey, and sent it to:
- RIPE OSS working group
- DNS-Ops IETF mailing list
- Posted on ISC’s social media
- We got 71 responses

Survey

https://ec.europa.eu/eusurvey/publication/RIPE88OpenSourceWGSurvey

When selecting an open source system to use for a critical application, how do you build confidence in the software project?			
		Answers	Ratio
documentation		45	63.380%
the user mailing list or forum is active and helpful		40	56.338%
releases are frequent/recent enough		39	54.930%
the software is already familiar to me		34	47.887%
versions are maintained for long enough		31	43.662%
there is more than one regular committer		25	35.211%
we conduct thorough testing of the software		25	35.211%
number of open, unresolved issues		16	22.535%
history of CVEs		16	22.535%
popularity (e.g. stars on GitHub)		14	19.718%
financial sponsors of the project are identified		12	16.901%
project test suite		9	12.676%
adequate packaging options		9	12.676%
software development process		8	11.268%
published roadmap		5	7.042%
badges on the project's homepage (example)		1	1.408%

BIND 9

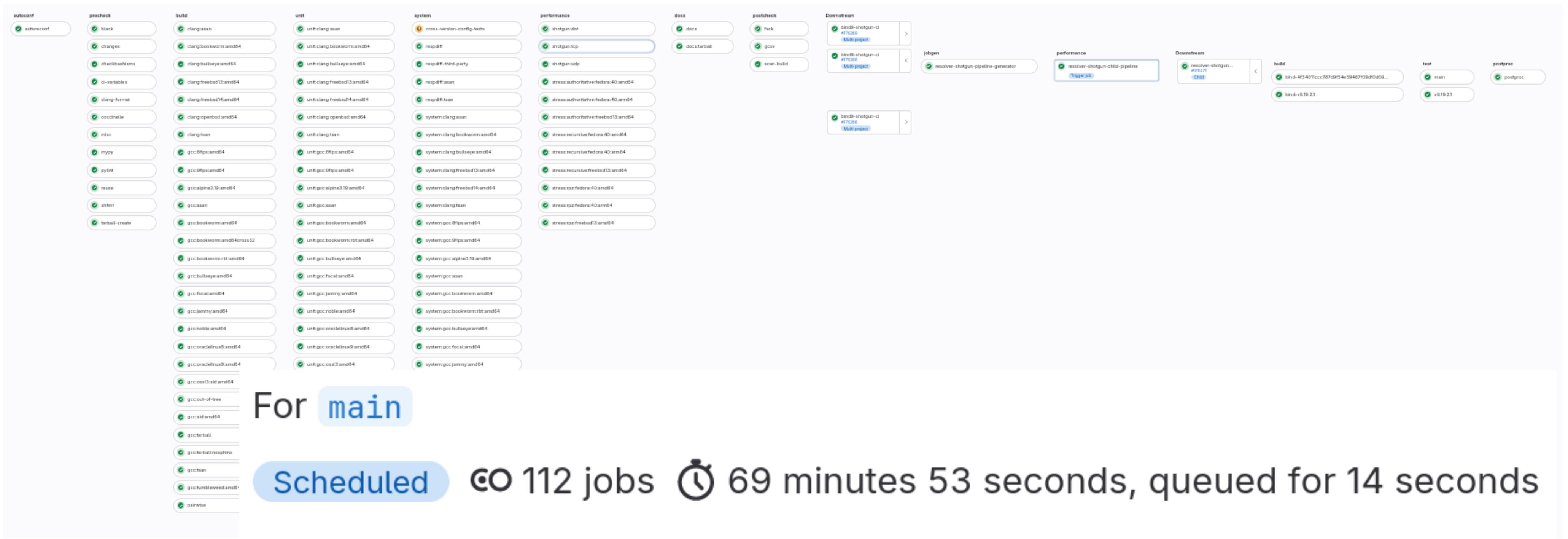
A DNS server – example OSS project

`https://gitlab.isc.org/isc-projects/bind9`

BIND 9 key quality processes

- Development process (commit requirements, coding standards, peer review)
- Test suite, test coverage, automated tools
- Ad-hoc and performance testing (real data)
- Investigation of all reported security issues (very time-consuming)

Continuous integration

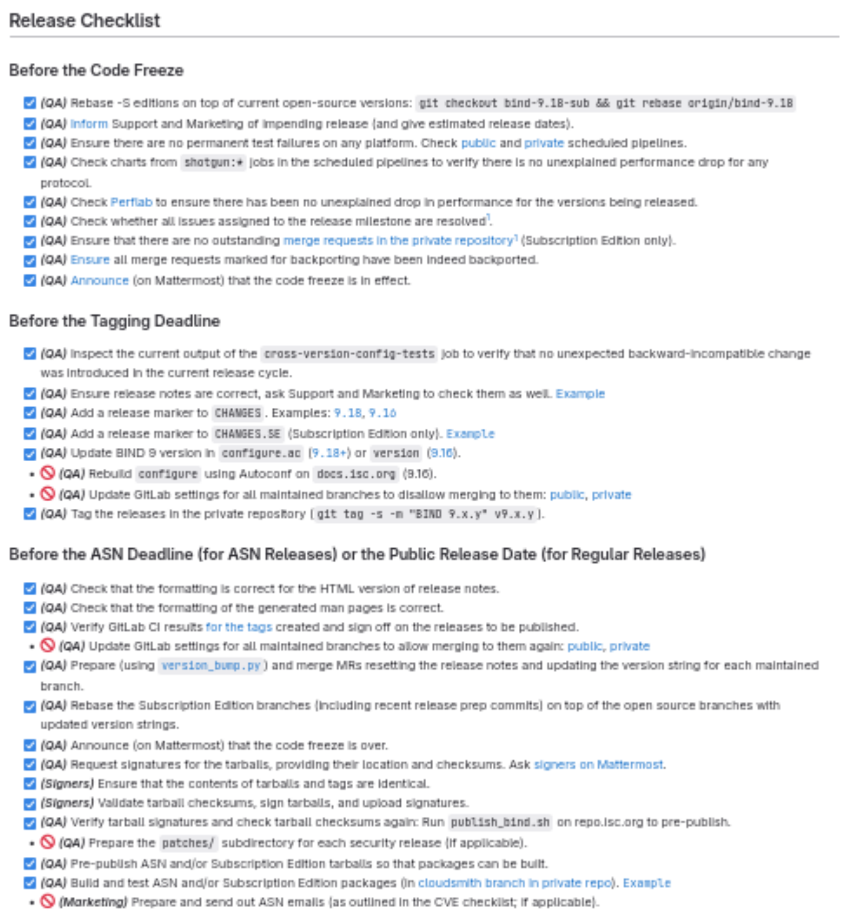


For `main`

Scheduled 112 jobs 69 minutes 53 seconds, queued for 14 seconds

Pipeline Needs Jobs 112 Failed Jobs 1 Tests 6321

BIND 9 Release process



Release Checklist

Before the Code Freeze

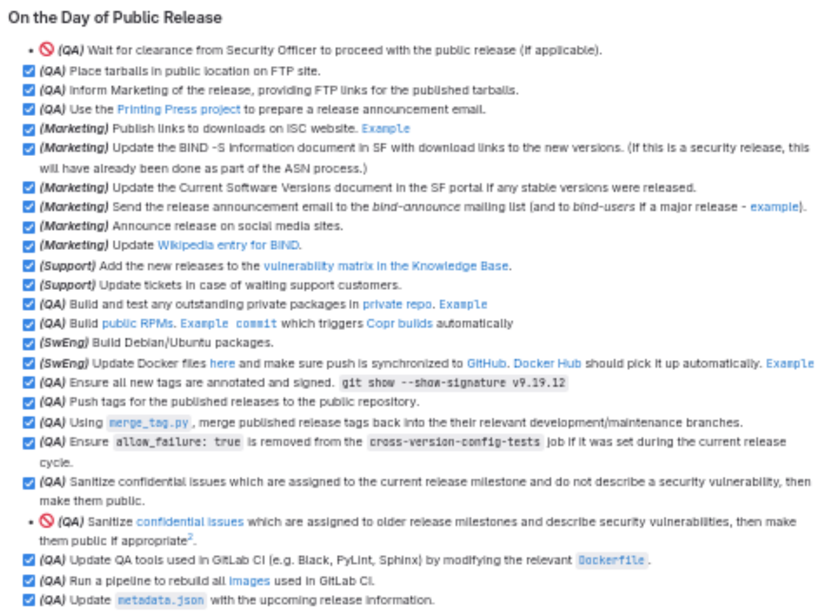
- [x] *(QA)* Rebase -S editions on top of current open-source versions: `git checkout bind-9.18-sub && git rebase origin/bind-9.18`
- [x] *(QA)* Inform Support and Marketing of impending release (and give estimated release dates).
- [x] *(QA)* Ensure there are no permanent test failures on any platform. Check public and private scheduled pipelines.
- [x] *(QA)* Check charts from `shotgun:*` jobs in the scheduled pipelines to verify there is no unexplained performance drop for any protocol.
- [x] *(QA)* Check Perflab to ensure there has been no unexplained drop in performance for the versions being released.
- [x] *(QA)* Check whether all issues assigned to the release milestone are resolved[1].
- [x] *(QA)* Ensure that there are no outstanding merge requests in the private repository[1] (Subscription Edition only).
- [x] *(QA)* Ensure all merge requests marked for backporting have been indeed backported.
- [x] *(QA)* Announce (on Mattermost) that the code freeze is in effect.

Before the Tagging Deadline

- [x] *(QA)* Inspect the current output of the `cross-version-config-tests` job to verify that no unexpected backward-incompatible change was introduced in the current release cycle.
- [x] *(QA)* Ensure release notes are correct, ask Support and Marketing to check them as well. Example
- [x] *(QA)* Add a release marker to `CHANGES`. Examples: 9.18, 9.16
- [x] *(QA)* Add a release marker to `CHANGES.SE` (Subscription Edition only). Example
- [x] *(QA)* Update BIND 9 version in `configure.ac` (9.18+) or `version` (9.16).
- 🚫 *(QA)* Rebuild `configure` using Autoconf on `docs.isc.org` (9.16).
- 🚫 *(QA)* Update GitLab settings for all maintained branches to disallow merging to them: public, private
- [x] *(QA)* Tag the releases in the private repository (`git tag -s -m "BIND 9.x.y" v9.x.y`).

Before the ASN Deadline (for ASN Releases) or the Public Release Date (for Regular Releases)

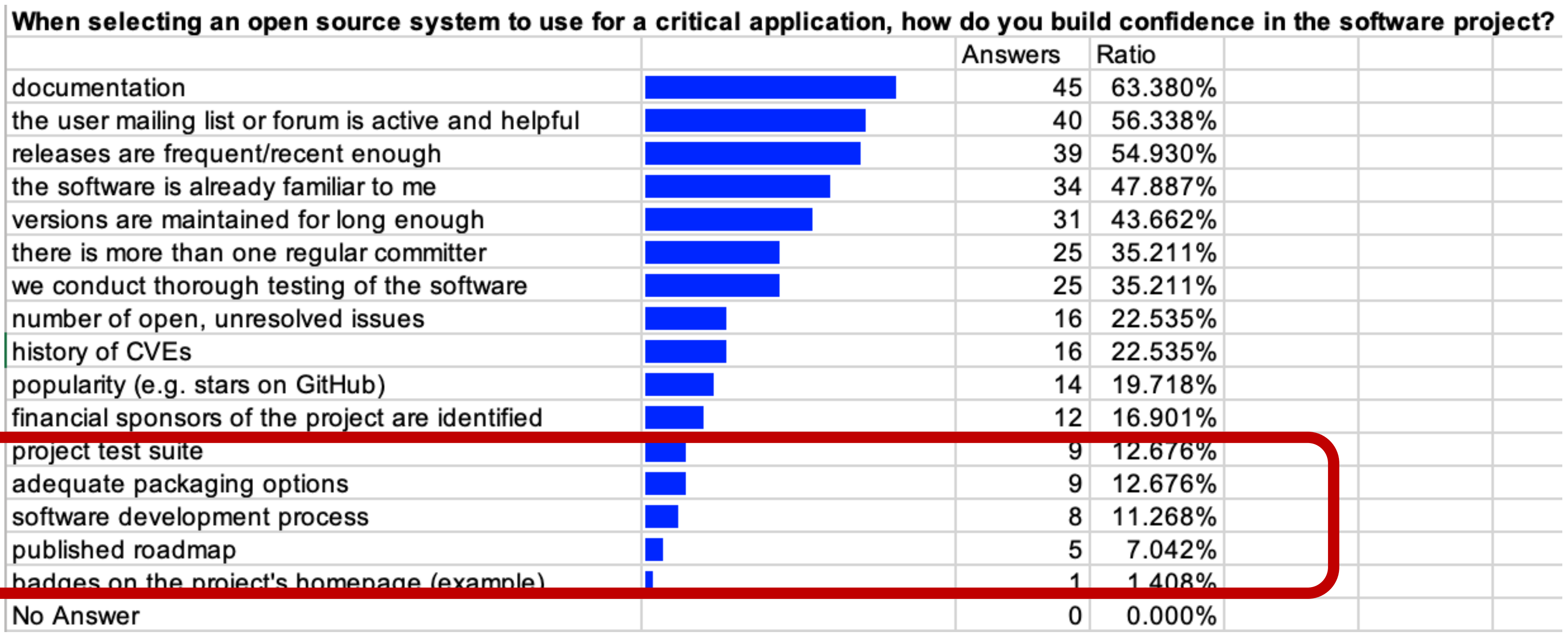
- [x] *(QA)* Check that the formatting is correct for the HTML version of release notes.
- [x] *(QA)* Check that the formatting of the generated man pages is correct.
- [x] *(QA)* Verify GitLab CI results for the tags created and sign off on the releases to be published.
- 🚫 *(QA)* Update GitLab settings for all maintained branches to allow merging to them again: public, private
- [x] *(QA)* Prepare (using `version_bump.py`) and merge MRs resetting the release notes and updating the version string for each maintained branch.
- [x] *(QA)* Rebase the Subscription Edition branches (including recent release prep commits) on top of the open source branches with updated version strings.
- [x] *(QA)* Announce (on Mattermost) that the code freeze is over.
- [x] *(QA)* Request signatures for the tarballs, providing their location and checksums. Ask signers on Mattermost.
- [x] *(Signers)* Ensure that the contents of tarballs and tags are identical.
- [x] *(Signers)* Validate tarball checksums, sign tarballs, and upload signatures.
- [x] *(QA)* Verify tarball signatures and check tarball checksums again: Run `publish_bind.sh` on repo.isc.org to pre-publish.
- 🚫 *(QA)* Prepare the `patches/` subdirectory for each security release (if applicable).
- [x] *(QA)* Pre-publish ASN and/or Subscription Edition tarballs so that packages can be built.
- [x] *(QA)* Build and test ASN and/or Subscription Edition packages (in cloudsmith branch in private repo). Example
- 🚫 *(Marketing)* Prepare and send out ASN emails (as outlined in the CVE checklist; if applicable).



On the Day of Public Release

- 🚫 *(QA)* Wait for clearance from Security Officer to proceed with the public release (if applicable).
- [x] *(QA)* Place tarballs in public location on FTP site.
- [x] *(QA)* Inform Marketing of the release, providing FTP links for the published tarballs.
- [x] *(QA)* Use the Printing Press project to prepare a release announcement email.
- [x] *(Marketing)* Publish links to downloads on ISC website. Example
- [x] *(Marketing)* Update the BIND -S information document in SF with download links to the new versions. (If this is a security release, this will have already been done as part of the ASN process.)
- [x] *(Marketing)* Update the Current Software Versions document in the SF portal if any stable versions were released.
- [x] *(Marketing)* Send the release announcement email to the *bind-announce* mailing list (and to *bind-users* if a major release - example).
- [x] *(Marketing)* Announce release on social media sites.
- [x] *(Marketing)* Update Wikipedia entry for BIND.
- [x] *(Support)* Add the new releases to the vulnerability matrix in the Knowledge Base.
- [x] *(Support)* Update tickets in case of waiting support customers.
- [x] *(QA)* Build and test any outstanding private packages in private repo. Example
- [x] *(QA)* Build public RPMs. Example commit which triggers Copr builds automatically
- [x] *(SwEng)* Build Debian/Ubuntu packages.
- [x] *(SwEng)* Update Docker files here and make sure push is synchronized to GitHub. Docker Hub should pick it up automatically. Example
- [x] *(QA)* Ensure all new tags are annotated and signed. `git show --show-signature v9.19.12`
- [x] *(QA)* Push tags for the published releases to the public repository.
- [x] *(QA)* Using `merge_tag.py`, merge published release tags back into the their relevant development/maintenance branches.
- [x] *(QA)* Ensure `allow_failure: true` is removed from the `cross-version-config-tests` job if it was set during the current release cycle.
- [x] *(QA)* Sanitize confidential issues which are assigned to the current release milestone and do not describe a security vulnerability, then make them public.
- 🚫 *(QA)* Sanitize confidential issues which are assigned to older release milestones and describe security vulnerabilities, then make them public if appropriate[2].
- [x] *(QA)* Update QA tools used in GitLab CI (e.g. Black, PyLint, Sphinx) by modifying the relevant `Dockerfile`.
- [x] *(QA)* Run a pipeline to rebuild all images used in GitLab CI.
- [x] *(QA)* Update `metadata.json` with the upcoming release information.

What makes a project trustworthy?



When selecting an open source system to use for a critical application, how do you build confidence in the software project?

	Answers	Ratio
documentation	45	63.380%
the user mailing list or forum is active and helpful	40	56.338%
releases are frequent/recent enough	39	54.930%
the software is already familiar to me	34	47.887%
versions are maintained for long enough	31	43.662%
there is more than one regular committer	25	35.211%
we conduct thorough testing of the software	25	35.211%
number of open, unresolved issues	16	22.535%
history of CVEs	16	22.535%
popularity (e.g. stars on GitHub)	14	19.718%
financial sponsors of the project are identified	12	16.901%
project test suite	9	12.676%
adequate packaging options	9	12.676%
software development process	8	11.268%
published roadmap	5	7.042%
badges on the project's homepage (example)	1	1.408%
No Answer	0	0.000%

NANOG™

Self-imposed policies

- Coding & review procedures
- OpenSSF software quality badge openssf best practices passing
  - Lots of specific quality process requirements, many reflected in the new government requirements
- ISC software defect and security vulnerability disclosure
- ISC CVSS scoring guidelines
- A lot of invisible work, but well-aligned with BCP

BIND9 practices vs. survey

	Our priority	Survey priority
CI & automated tests	#1	# 14
code reviews & standards	#1	# 13
Documentation	#?	# 1

Conclusion

- Are users, or the experts, wrong?
- Are users assessing results, whereas experts are focused on processes?
- Is all the preoccupation with software security missing a more fundamental problem?
- Is this some kind of learned helplessness on the part of users, who may just be overwhelmed?
- Or ....

NANOG™

Thank you

https://www.isc.org

vicky@isc.org