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 ...

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



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





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%



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



7

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

autoconf	precheck	build	unit	system	performance	docs	postcheck	Downatream						
e sutorecont	o black	Cangasan	🖉 unit:clangrasan	eross-version-config-tests	S shotgunidot	Ø doca	@ hck	bind8-shotpun-cl atrocce Wu8-project						
	C changes	clang/bookworm.amd64	unit:clang:bookworm:amd84	C respd#	💿 shotguntop	ladratzob 🧿	@ goov	binds-shotpun-ci	jobpen	performance	Downstream			
	C checkbaahiama	clangibullusystamd64	🕑 unit:clang:bullaeyecamd64	respd#-third-party	💿 shotgunudp		🔕 scan-build	Milliproject)	encliver-shotgun-pipeline-generator	o resolver-shotgun-child-pipeline	resolver-shotgun	build	test	postproc
	C d-variables	clangstreebsd13:smd84	unit:clang/freebsd13:amd84	💿 respdittasan	stress sufficitative:fedora:40 amd64					(Trigger (st)	(CRAR)	Divid-46340Ttccc787d9454e59467702d40d08	O nah	peatproc
	Camp-format	Clangtheebsd14tamd64	unit:clang/freebsd14:amd64	e respdittsan	streax authoritative:fedora:40 arm84							@ bind-v939.23	● +919.23	
	Coccinette	clangropenbud send64	@ unit:clang:openbod:amd64	systemictang asan	stress suffortistive:treebsd13.amd64			552011A						
	o misc	Cangtuan	o unit:clangitian	systemcclang bookwormcand64	atress recursive fedors: 40: amd64			Well-project.						
	💿 туру	gcc:8fps:amd64	O unit:gcc.81px and64	o systemcclang.bullaeyecamd54	atress recursive fedors: 40: arm64									
	O pytint	gcc:9/pscamd84	unit:gcc:9flps:amd64	system clang freebadt3:arrid04	stress:recursive:freebadl3:ared64									
		gcc:alpine3.10:amd64	unt:gccalpine3.19 amd64	system clang freebad14cand04	Ø stress.rpz.fedora:40.amd84									
	intert O	gecaan	@ untrocasan	systemclang.tean	🖉 stress.rpz.fedora.40.arm84									
	Co tarbali-create	gcc:bookworm:and64	@ unit:gcc.bookworrcand64	system:goc:8fips.amd84	stress:rpz freebud13 arnd04									
		gcc:bookworm:amd64cross32	o unit:gcc.bookworm:rbt.and64	System: poc.9/(ps.amd84										
		gcc:bookworm/bt.amd64	@ unit:gcc.bullasyscand64	system:goc:s/pine3.19.amd64										
		grabulaeyeamd64	@ unit:gccfocalamd84	system:goc:asan										
		gcc:focalamo84	@ unt:gccjammyamitt4	system:goc.bookworm.and64										
		gcc:jammytamd84	@ unit:gcc:noblecamd64	system:goc.bookworm:rbt.amd64										
		gcc:nobie.amd54	👩 unit:gcc.oraclelin.oit.amd64	o system god bullaeye amdt4										
		gcc:oracielinus@carndt4	🖉 unit:gcc:oraclelinuxit:amd64	g system god focal amd64										
		gccoraclelinusRamd54	o untrgccossi3amd64	g system.goc.jammy.amd64										
		gcctarball gcctarball gcctarball	or main Schedul		112 jobs	Ō	69 m	ninutes	s 53 sec	onds, d	queue	ed for 14 s	seco	nds
	0	NAN	Pipeliı	ne N	eeds	Job	s 11:	2	Failed J	obs 1	Te	ests 632	1	

BIND 9 Release process

Release Checklist

Before the Code Freeze

- 🗹 (QA) Rebase -S editions on top of current open-source versions: git checkout bind-9.18-sub 🎄 git rebase origin/bind-9.18
- (QA) inform Support and Marketing of Impending release (and give estimated release dates).
- (QA) Ensure there are no permanent test failures on any platform. Check public and private scheduled pipelines.

(QA) Check charts from shotgun:* jobs in the scheduled pipelines to verify there is no unexplained performance drop for any Incotocol

- (QA) Check Perflab to ensure there has been no unexplained drop in performance for the versions being released.
- (QA) Check whether all issues assigned to the release milestone are resolved¹
- (QA) Ensure that there are no outstanding merge requests in the private repository¹ (Subscription Edition only).
- (QA) Ensure all merge requests marked for backporting have been indeed backported.
- (QA) Announce (on Mattermost) that the code freeze is in effect.

Before the Tagging Deadline

- (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.
- (QA) Ensure release notes are correct, ask Support and Marketing to check them as well. Example
- (QA) Add a release marker to CHANGES. Examples: 9.18, 9.10
- (QA) Add a release marker to CHANGES.SE (Subscription Edition only). Example
- (QA) Update BIND 9 version in configure.ac (9.18+) or version (9.16).
- O (QA) Rebuild configure using Autoconf on docs.isc.org (9.16).
- O (QA) Update GitLab settings for all maintained branches to disallow merging to them: public, private
- QA) Tag the releases in the private repository (git tag -s -n "BIND 9.x.y" v9.x.y).

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

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

On the Day of Public Release

- Q (QA) Wait for clearance from Security Officer to proceed with the public release (if applicable).
- (QA) Place tarballs in public location on FTP site.
- (QA) Inform Marketing of the release, providing FTP links for the published tarballs.
- (QA) Use the Printing Press project to prepare a release announcement email.
- Marketing) Publish links to downloads on ISC website. Example
- (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.)
- (Marketing) Update the Current Software Versions document in the SF portal if any stable versions were released.
- (Marketing) Send the release announcement email to the bind-announce mailing list (and to bind-users if a major release example).
- (Marketing) Announce release on social media sites.
- (Marketing) Update Wikipedia entry for BIND.
- Support) Add the new releases to the vulnerability matrix in the Knowledge Base.
- (Support) Update tickets in case of waiting support customers.
- (QA) Build and test any outstanding private packages in private repo. Example
- (QA) Build public RPMs, Example connit which triggers Copr builds automatically
- (SwEng) Build Deblan/Ubuntu packages.
- (SwEng) Update Docker files here and make sure push is synchronized to GitHub. Docker Hub should pick it up automatically. Example
- (QA) Ensure all new tags are annotated and signed. git show --show-signature v9.19.12
- (QA) Push tags for the published releases to the public repository.
- (QA) Using introe_tag.py, merge published release tags back into the their relevant development/maintenance branches.
- (QA) Ensure allow_failure: true is removed from the cross-version-config-tests job if it was set during the current release
- cycle.
- (QA) Sanitize confidential issues which are assigned to the current release milestone and do not describe a security vulnerability, then make them public.
- Q(QA) Sanitize confidential issues which are assigned to older release milestones and describe security vulnerabilities, then make them public if appropriate².
- (QA) Update QA tools used in GitLab CI (e.g. Black, PyLint, Sphinx) by modifying the relevant Bookerfile.
- (QA) Run a pipeline to rebuild all images used in GitLab CI.
- (QA) Update netadata.json with the upcoming release information.

10

What makes a project trustworthy?

When selecting an open source	system to use for a critica	l application, how do	vou build confidence in	the software project?
The concerning an open ocures		approation, non ao	jea sana comachec m	

	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%	
he software is already familiar to me	34	47.887%	
versions are maintained for long enough	31	43.662%	
here is more than one regular committer	25	35.211%	
ve conduct thorough testing of the software	25	35.211%	
number of open, unresolved issues	16	22.535%	
nistory of CVEs	16	22.535%	
oopularity (e.g. stars on GitHub)	14	19.718%	
inancial sponsors of the project are identified	12	16.901%	
roject test suite	9	12.676%	
dequate packaging options	9	12.676%	
oftware development process	8	11.268%	
oublished roadmap	5	7.042%	
padges on the project's homenage (example)	1	1 408%	
lo Answer	0	0.000%	



Self-imposed policies

- <u>Coding & review procedures</u>
- 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

🔷 N A N O G

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

へ N A N O G[™]

Thank you

https://www.isc.org vicky@isc.org