AT COMPUTING

ATYPICAL OPEN SOURCE GURUS

CHECKLIST OPEN SOURCE SOFTWARE

Analyse-tool voor OSS projecten

Checklist Open Source Software

Een gebalanceerde scorecard voor selectie, adoptie en risicobeheersing van open source software

Introductie

Naar schatting 70-90% van alle software wereldwijd is open source software (OSS).* Toch zijn veel organisaties terughoudend met het omarmen van OSS, niet zelden vanwege een gebrek aan kennis en/of gedegen, onafhankelijk advies.

Vaak klinkt het argument: "We willen een leverancier met een contract, anders dragen we te veel verantwoordelijkheid." Dat klinkt logisch, maar het berust op een misverstand over hoe risico's in softwareketens werkelijk werken.

Een contract biedt namelijk geen veiligheid — hooguit afspraken over responstijden, werkzaamheden of correctieve acties. Bovendien geeft een contract geen enkel inzicht in de kwaliteit van de software zelf. Bij closed-source oplossingen is dat zelfs onmogelijk: alleen de leverancier weet wat er in de code gebeurt. Je vertrouwt per definitie op de blauwe ogen van de leverancier.

De praktijk laat bovendien zien dat ook grote leveranciers met SLA's en miljoenenbudgetten niet immuun zijn voor fouten of incidenten. Denk aan langdurige cloudstoringen, grote security-incidenten, kritieke kwetsbaarheden in closed-source producten of fouten in automatisering die wereldwijd impact hebben. Formaliteit vervangt geen realiteit.

De Nederlands/Australische veiligheidsexpert **Sidney Dekker** beschrijft dit als het verschil tussen work-as-imagined (hoe we denken dat het werk gaat via beleid en contracten) en work-as-done (de daadwerkelijke praktijk). In de praktijk komt het neer op één simpele waarheid:

> *Je kunt verantwoordelijkheid niet uitbesteden. Je kunt hooguit de verantwoordelijkheid voor bepaalde taken elders beleggen.*

Of je nu open source of c software gebruikt: jij en/of je organisatie blijft uiteindelijk verantwoordelijk voor de stabiliteit, beschikbaarheid, integriteit, veiligheid (BIV) en continuïteit van de systemen waarop je vertrouwt. Dat betekent dat volwassen open source-gebruik niet draait om het vinden van een leverancier, maar om het begrijpen van de kwaliteit van het project, de community en de risico's. En om het nemen van de juiste, proportionele maatregelen om de risico's te beheersen.

*bron: https://www.linuxfoundation.org/blog/blog/a-summary-of-census-ii-open-source-software-application-libraries-the-world-depends-on

Een andere kijk op controle en risicobeheersing

Bij AT Computing zijn we dol op OSS en verdienen we er al sinds 1985 onze boterham mee. Met deze checklist hopen we onze kennis en ervaring met open source projecten en OSS te delen en te helpen bij een verdere adoptie van OSS. Wij zijn ervan overtuigd dat je met de juiste afwegingen op een verantwoorde en betrouwbare manier de voordelen van OSS kunt benutten.

Voordat je direct begint met het invullen van de checklist, is het belangrijk om te beseffen dat het ecosysteem rondom een stuk OSS fundamenteel anders werkt dan die van gesloten (propriëtaire) software.

Bij propriëtaire software is de leverancier van de software ook de enige partij die weet hoe de software in elkaar zit en ook de enige die onderhoud aan de software kan doen. Dit heeft weliswaar als voordeel dat je al je zaken met een enkele partij kunt doen, maar daar kleven tegelijkertijd ook twee enorme nadelen aan:

- Je bent volledig afhankelijk van de leverancier (vendor lock-in) en/of aan de leverancier gelieerde partners
- Je hebt geen enkel zicht op wat zich onder de motorkap allemaal afspeelt (geen transparantie).

Bij OSS is dit totaal anders. Omdat de broncode van de software open is, kun je precies zien hoe de software in elkaar zit en werkt. Ook ben je voor onderhoud of verbeteringen niet (volledig) van de leverancier afhankelijk: je kunt eventueel zelf een verbetering schrijven voor het betreffende stuk software. In het meest extreme geval zou je zelf een kopie (fork) van de software kunnen maken en een geheel eigen versie verder ontwikkelen en onderhouden.

Ook is het bij OSS gebruikelijk dat de leverancier (vendor) niet de partij hoeft te zijn die onderhoud of ondersteuning niet hetzelfde hoeft te zijn. Dit is vergelijkbaar met onderhoud aan een auto doen bij een lokale vakgarage in plaats van bij de merkdealer.

Wil je meer weten over het beheersen van OSS risico's? Lees dan ook onze white paper hierover: Open Source Software Onder Controle

Waarom deze checklist?

Hoewel we als AT Computing groot fan zijn van OSS, kent OSS wel degelijk risico's. Deze risico's zijn niet groter dan die van propriëtaire software, maar wel anders. Om deze risico's inzichtelijk te hebben en te beheersen is een zekere vorm van onderzoek nodig. Je kunt dit een beetje zien als *due diligence* bij een bedrijfsovername of bij het selecteren van de leverancier van propriëtaire software.

Veel organisaties hebben wel kaders, richtlijnen en eisen voor propriëtaire software, maar missen een objectief en reproduceerbaar hulpmiddel om OSS te beoordelen. Keuzes worden daardoor vaak op onderbuikgevoel, aannames of gekleurde informatie gemaakt en niet op basis van gedegen analyse. Deze checklist brengt daar verandering in.

Met 50 concrete criteria, verdeeld over vijf domeinen, ontstaat een genuanceerd en reproduceerbaar beeld van de volwassenheid, kwaliteit en betrouwbaarheid van een project. Niet als exacte wetenschap, maar als hulpmiddel om gesprek, beleid en besluitvorming te objectiveren.

De checklist helpt je echter niet alleen met beoordelen, maar biedt -net zo belangrijk- diverse tips om in actie te komen! Op de laatste pagina's staat hoe je zelf kunt bijdragen aan een open source project. Zo help je niet alleen jezelf, maar ook anderen. De kracht van de community!

De checklist

Op basis van vijf categorieen, behandelen we de 50 belangrijkste criteria om een OSS project gedegen en genuanceerd te beoordelen. De checklist is zo opgezet dat deze toepasbaar is op zowel losse libraries, frameworks, tools, platforms, infrastructuur-componenten en complete software-producten.

Belangrijk: zie de criteria en scores niet als exacte wetenschap, maar als een leidraad. Het kan voorkomen dat een bepaald stuk OSS laag scoort op bepaalde punten, maar slechts een minimale rol speelt binnen het gehele landschap. De impact van dit zwakke component kan dan alsnog klein zijn, waardoor het geen serieuze bedreiging of risico vormt.

De criteria zijn vooral waardevol bij het uniform/eenduidig classificeren van het software-landschap en dragen daarmee bij aan het objectiveren van discussies over bijvoorbeeld risicobeheersing en bij strategische afwegingen. Hierdoor ben je beter in staat de juiste beslissingen te nemen en adequate maatregelen te treffen. Ook kan de checklist als gereedschap worden gebruikt om beleid op gebied van OSS mee te borgen in de dagelijkse operatie.

Hoe vul je de tabellen in?

Iedere tabel bevat 10 criteria, die ieder een gewicht hebben meegekregen. Belangrijke criteria hebben hierdoor meer invloed op de eindscore dan minder belangrijke criteria. Het maximale gewicht van een criterium is 3, het minimale gewicht is 1.

Het kan zijn dat je gedeeltelijk aan een criterium voldoet. Zo kan het bijvoorbeeld voorkomen dat een project wel een openbare roadmap heeft, maar dat de kwaliteit van deze roadmap ondermaats is of de inhoud gedateerd. In dat geval kun je ervoor kiezen om een deel van de punten toe te kennen. Ga hier als volgt mee om:

- **Voldoet het volledig?** -> ken de maximale score toe
- **Voldoet het voor een aanzienlijk deel** (bijv. >50%) -> ken max. gewicht - 1 toe (dus 2 ipv 3, 1 ipv 2 of 0 ipv 1)
- **Voldoet het minimaal of nauwelijks** -> ken een score van 1 toe bij max. gewicht 3 of 0 bij max. gewicht 1 of 2
- **Voldoet het in geheel niet?** -> ken een score van 0 toe

Is het antwoord onbekend, doe dan eerst een stuk onderzoek of vul 0 in. Onbekend is immers per definitie een risico!

Voorbeeld

max. 11 punten

Nr	Aspect	Gewicht	Score
1	Project heeft een duidelijke governance-structuur	3	*2*
2	Project heeft een openbare roadmap	2	*1*
3	Pull Requests worden tijdig gereviewd	2	*1*
4	Maintainers zijn vindbaar en benaderbaar	2	*2*
5	Codebase bevat geen ongebruikte of verouderde modules	1	*0*
	Totale score:		**6**

1. Projectkwaliteit & Governance

(max. 20 punten)

Nr	Aspect	Gewicht	Score
1	Project heeft een duidelijke governance-structuur (foundation / consortium / duidelijk eigenaarschap)	3	
2	Er is een actieve en herkenbare maintainer-groep	3	
3	Project heeft een openbare roadmap	2	
4	Project heeft een formele release-strategie (semver, release notes)	2	
5	Licentie is helder, open en juridisch veilig (OSI-approved)	3	
6	Licentiewijzigingen in de laatste 24 maanden? (geen = positief)	2	
7	Project is onafhankelijk van één commerciële partij	2	
8	Er is een Code of Conduct	1	
9	Er is een Security Policy / Responsible Disclosure richtlijn	1	
10	Projectwebsite en documentatie zijn professioneel onderhouden	1	
	Totale score:		

2. Community & Activiteit

(max. 20 punten)

Nr	Aspect	Gewicht	Score
1	Aantal actieve contributors per maand is stabiel of groeiend	3	
2	Updates in de laatste 90 dagen (levendig project)	3	
3	Issues worden opgepakt en beantwoord	2	
4	Pull Requests worden tijdig gereviewd	2	
5	Community is vriendelijk, open en reageert constructief	2	
6	Er is engagement buiten GitHub (Slack, Forum, Mastodon, Discord)	1	
7	Project heeft een duidelijke bijdrage-richtlijn (CONTRIBUTING.md)	2	
8	Maintainers zijn vindbaar en benaderbaar	2	
9	Er bestaan (grote) gebruikers of referenties in bedrijfscontext	2	
10	Geen extreme afhankelijkheid van één persoon (bus-factor)	1	
	Totale score:		

3. Codekwaliteit, Security & Testdekking

(max. 20 punten)

Nr	Aspect	Gewicht	Score
1	Project bevat geautomatiseerde tests (unit/integration)	3	
2	Testdekking is aantoonbaar (coverage >= 70% idealiter)	2	
3	CI/CD pipelines aanwezig (linting, builds, security scans)	3	
4	Dependency management is up-to-date	2	
5	Security-audits aanwezig (formeel, community-gedreven of extern betaald)	3	
6	Kwetsbaarheden worden tijdig gepatcht	2	
7	Code is gestructureerd, leesbaar en volgt conventies	1	
8	Backwards compatibiliteit wordt bewaakt	1	
9	Codebase bevat geen ongebruikte of verouderde modules	1	
10	Er is threat-modeling / security-architectuur gedocumenteerd	2	
	Totale score:		

4. Functionaliteit, Volwassenheid & Roadmap

(max. 20 punten)

Nr	Aspect	Gewicht	Score
1	Functionaliteit is stabiel en volwassen	3	
2	Project wordt gebruikt in productieomgevingen	3	
3	Releasefrequentie is consistent	2	
4	Compatibiliteit met platformen/versies is duidelijk	2	
5	Er is backward compatibility tussen releases	2	
6	Roadmap is realistisch en actief bijgewerkt	2	
7	Architectuur is modern en modulair	2	
8	Migratiegidsen of upgrade-instructies beschikbaar	1	
9	Documentatie beschrijft limieten, risico's en edge cases	1	
10	Project heeft een ecosysteem (plugins, integraties, tooling)	2	
	Totale score:		

5. Documentatie, Support & Professionalisering

(max. 20 punten)

Nr	Aspect	Gewicht	Score
1	Documentatie volledig, gebruiksvriendelijk en actueel	3	
2	Installatiehandleidingen helder en compleet	2	
3	Troubleshooting-sectie aanwezig	1	
4	API-documentatie aanwezig en accuraat	2	
5	Tutorials, voorbeelden of demo's beschikbaar	2	
6	Betaalde ondersteuning beschikbaar (optioneel)	1	
7	Er bestaan trainingen, boeken of courses	1	
8	Issue-templates en bug-reportingrichtlijnen beschikbaar	1	
9	Er is een duidelijk release- en changelogbeleid	2	
10	Documentatie bevat security-aanbevelingen en best practices	2	
	Totale score:		

Totaalscore

(max. 100 punten)

Domein	Max	Score
Projectkwaliteit & Governance	20	
Community & Activiteit	20	
Codekwaliteit & Security	20	
Functionaliteit & Maturiteit	20	
Documentatie & Support	20	
Totaalscore	**100**	

Conclusie op basis van eindscore

	Eindscore	Conclusie
🟢	>75 punten	Dit project is (zeer) volwassen en klaar voor gebruik!
🟡	50-75 punten	Er zijn aandachtspunten die wellicht aandacht vereisen voor gebruik.
🟠	25-50 punten	Er zijn serieuze tekortkomingen. Kijk wat je hieraan kunt doen of zoek een ander project.
🔴	<25 punten	Het project is momenteel ongeschikt voor gebruik. Overweeg om je bevindingen met het project te delen om ze zo te helpen.

Hoe kun je als organisatie bijdragen aan een open-source project?

Veel organisaties beoordelen open source projecten puur als consument, maar een open source project staat of valt bij samenwerking en bijdragen vanuit de *community*. Het mooie hiervan is dat iedereen onderdeel van die community kan zijn, oftewel: als een open source project veel waarde voor je levert, maar je toch een aantal risico's ziet dan kun je in veel gevallen actie ondernemen en daar wat aan doen. In dit hoofdstuk vind je een aantal dingen die je praktisch kunt doen om bij te dragen aan een open source project. Op die manier help je niet alleen je eigen risico's te mitigeren of verkleinen, maar ook die van andere gebruikers van het project. Dat noemen we nog eens een win-win!

1. Verhoog de testdekking en codekwaliteit

Dit is wellicht de meest *klassieke* manier van bijdragen aan een open source project: code schrijven!

- **Schrijf tests**: unit- of integratietests kunnen de softwarekwaliteit aanzienlijk verbeteren
- Ontwerp/implementeer **CI-pipelines** om automatische tests te faciliteren
- Los technische issues of achterstallig onderhoud op (pak **issues, merge of feature requests** op)
- Verhelp **kwetsbaarheden** of upgrade **dependencies**

Deze bijdrage versterkt de stabiliteit én het vertrouwen in het project.

2. Verbeter documentatie (vaak onderschatte bijdrage)

- Voeg betere **installatieinstructies** toe
- Maak **tutorials** of voorbeelden
- **Documenteer** edge cases
- Documenteer **API's** of de interne **architectuur**

Documentatie is vaak een zwak punt van open-source software — hier kun je relatief snel veel impact maken.

3. Betaal voor audits of security reviews

Je kunt bijdragen door bijvoorbeeld:

- Een externe **security-audit** te laten uitvoeren (en de resultaten hiervan te delen met maintainers van het project)
- Meehelpen om urgente **security-patches** te financieren
- Bijdragen aan **threat-modeling** of een pentest

Dit is extreem waardevol voor projecten die security-kritiek zijn.

4. Doneer geld of sponsor maintainers

Veel open-source projecten draaien op basis van vrijwilligers. Met sponsoring kun je:

- **Features** laten ontwikkelen
- Security-gerichte **releases financieren**
- Maintainer **burnout voorkomen**
- **Stabiliteit** en continuïteit waarborgen
- **Naamsbekendheid** van het project vergroten omdat er budget is voor bijvoorbeeld marketing/promotie
- **Reis- en verblijfskosten** dekken voor samenwerking of bezoek van (community) events door projectleden

Het is opvallend dat bedrijven zonder problemen veel geld betalen aan licenties of SLA support, maar weinig bereidheid tonen om te doneren of op andere wijze een financiële bijdrage te leveren aan open source projecten waar ze volop (commercieel) gebruik van maken zonder verdere kosten.

5. Neem actief deel aan community en governance

- Organiseer **meetups** of kennissessies over de software/het project
- Help bij (modereren van) **discussies** of RFC's
- Wordt onderdeel van de **governance** (als contributor, co-maintainer, committer, reviewer)
- **Sponsor infrastructuur**: CI/CD pipelines (bijvoorbeeld door kosten voor GitHub of GitLab te vergoeden), mirror servers, hosting

6. Bied langdurige betrokkenheid

Projecten bloeien door consistentie. Als bedrijf kun je structureel bijdragen door:

- Minimaal 1 dag per maand **developer-tijd** te reserveren
- SLA-achtige **steun** te geven aan maintainers
- **Feature-roadmaps** af te stemmen
- Te helpen met **promotie / endorsement** van het project (blogs, use cases, social media campagne)

Tot slot

Open source software kan veilig en verantwoord worden gebruikt, mits dit weloverwogen gebeurt. Met deze checklist heb je niet alleen een gedegen instrument in handen om een weloverwogen besluit te nemen, maar heb je met de bijdrage-suggesties ook gelijk een set van maatregelen te pakken die kunnen helpen om jouw risico's te mitigeren en reduceren. Op die manier maak je het open source ecosysteem sterker en daar plukt uiteindelijk iedereen de vruchten van.

Contact & Services

AT Computing is vooral bekend vanwege haar trainingen, maar staat ook sinds 1985 klaar voor het leveren van diepgaande kennis over het gebruik van open source software binnen een IT-omgeving. Vanuit technisch perspectief kunnen we designs, configuratiescripts, infrastructuur of Python code reviewen, advies geven bij IT-uitdagingen of helpen bij het opzetten van een gestroomlijnde, geautomatiseerde workflow met bijvoorbeeld Ansible, Terraform, GitLab of Kubernetes. Ook voor advies over lokale LLM's voor GenAI toepassingen hebben wij expertise in huis. Vanuit governance- en risicomanagement perspectief kunnen we helpen bij het opstellen van open source software-beleid, advies geven over IT security, risicobeheersing en u helpen compliant te blijven zonder de voordelen van open source software op te geven.

Contactgegevens

AT Computing B.V.
Arnhemsestraatweg 33
6881ND Velp
The Netherlands

Email: info@atcomputing.nl
Phone: +31 (0)24 352 72 82
Web: https://atcomputing.nl