



Open Source Security

Een advies voor veilig gebruik van open source software

Aanleiding voor dit advies

Open source software (OSS) heeft de afgelopen decennia wereldwijd een belangrijke stempeel gedrukt op de ontwikkeling van digitale dienstverlening. En de interesse om OSS nog breder in te zetten blijft groeien. Dit geldt ook voor de overheid die zichzelf tot doel heeft gesteld om meer OSS in te zetten en zelf ontwikkelde software als open source vrij te geven via het 'open, tenzij' principe.

Die keuze om het gebruik van open source binnen de overheid verder te stimuleren is gebaseerd op een aantal overwegingen. Zo moet de inzet van OSS onder andere de afhankelijkheid van commerciële IT-leveranciers beperken, de risico's van *vendor lock-in* verkleinen, het innovatief vermogen versterken, bijdragen aan een meer open en transparante digitale overheid en ertoe leiden dat IT-voorzieningen die bekostigd worden met publieke middelen ook vervolgens voor eenieder (her)bruikbaar zijn.

De risico's en uitdagingen van open source Het ontwikkelen, en in zekere zin ook het inzetten, van OSS kent tegelijkertijd wel een andere dynamiek en daarmee ook andere uitdagingen dan die gelden voor meer 'reguliere' commercieel verkrijgbare IT-diensten, waar de overheid en private partijen vooral ervaring mee hebben. Een bredere inzet van OSS vraagt daarmee om andere werkwijzen en andere beheerafwegingen dan welke gebruikersorganisaties veelal gewend zijn en kennen.

Eén van de punten waar de inzet en ontwikkeling van OSS verschilt van die van commerciële aanbieders is het risicomanagement. OSS kent enkele andere risico's en vraagt op een aantal punten om ander risicomanagement dan organisaties gewend zijn met closed

source software (CSS) – onder andere ook ten aanzien van security. Daar gaat dit stuk over.

Open source software; veiliger dan closed source?

De ambitie om (overheidsbreed) meer OSS in te willen zetten, steunt, naast de hierboven genoemde overwegingen, ook op de veronderstelling dat OSS veiliger is dan CSS omdat – zo is de aanname – eenieder die dat wil de source code van OSS kan inzien, kan inspecteren en zo nodig kan (laten) verbeteren mochten hierin kwetsbaarheden worden aangetroffen. Het open, transparante karakter van OSS zou zo moeten leiden tot het voorkomen dan wel sneller bekend worden van kwetsbaarheden, een sneller herstel daarvan en zo uiteindelijk tot een veiliger softwareproduct.

Uit (wetenschappelijk) onderzoek blijkt vooralsnog weinig concrete steun voor de hypothese dat OSS (inherent) veiliger is dan CSS. Diverse onderzoeken¹ laten zien dat er in de praktijk weinig verschil zit in het aantal kwetsbaarheden tussen enerzijds OSS en anderzijds CSS. De kwaliteit van de code lijkt vooral te worden bepaald door professionaliteit van het ontwikkelteam en de manier waarop zij code ontwikkelen, los van of dit OSS of CSS is. Uit recent onderzoek blijkt dat het overgrote deel van de OSS ten minste één of meerdere bekende kwetsbaarheden bevat; de helft bevat één of meer kwetsbaarheden waarvan bekend is dat die actief misbruikt (kunnen) worden. Het gemiddelde aantal kritieke kwetsbaarheden per OS-project varieert tussen de 2.6 voor .Net codes en 9.5 voor OSS in Java. Het totale aantal kwetsbaarheden per OS-project, inclusief de minder ernstige, ligt een stuk hoger, namelijk op gemiddeld 23 voor OS in .Net en 90 voor OS in Java.

¹ Fagundez et al, 2018, A literature review about the difference in security for open source and proprietary source software – and its influence in Open Science, Proceedings Mere 2018 Conference, Barcelona.

Schryen, 2011, Is Open Source Security a Myth. CACM, 54:5, 130-139.

aan voor eenieder die die software wil gebruiken. De keerzijde hiervan is echter ook een zekere vrijblijvendheid. Gebruikers kunnen er namelijk niet op rekenen; er niet vanuit gaan, dat de betreffende OSS gedurende de gehele gebruiksperiode adequaat ondersteund en onderhouden blijft, en dat kwetsbaarheden en andere securityproblemen altijd tijdig verholpen en gepatched worden. Er kunnen zich forse problemen voordoen in een situatie waarin een bedrijf of overheidsorganisatie een bepaalde OSS-voorziening heeft geïmplementeerd voor een kritiek bedrijfsproces, om er vervolgens achter te komen dat de ontwikkel- en beheer community die zich over de OSS had ontfermd, uit elkaar is gevallen en eigenlijk weinig tijd meer steekt in het goede beheer en tijdig patchen van die OSS. De organisatie moet vervolgens zo snel mogelijk op zoek naar een alternatief voor de OSS-voorziening en deze, vanwege o.a. security risico's die voortkomen uit gebrekkig of onzeker onderhoud, snel uitfaseren.

Het ontbreken van garanties, van de zekerheid dat OSS goed wordt onderhouden; dat er geen enkele partij echt aanspreekbaar is óp, dan wel aansprakelijk is vóór, adequaat onderhoud, maakt dat veel professionele organisaties terughoudend zijn met het gebruik van OSS, zeker in kritieke bedrijfsprocessen. De mate waarin dit risico speelt hangt af van de robuustheid, omvang en professionaliteit van de betrokken OSS community en of er bijvoorbeeld een partij of bedrijf is die zich op enigerlei wijze heeft ontfermd over de betreffende OSS ontwikkeling. Uit recent [onderzoek van de Linux Foundation](#) blijkt bijvoorbeeld dat bij een aanzienlijk deel van de OSS projecten de software wordt ontwikkeld en beheerd door een kleine groep, en soms slechts één of enkele, ontwikkelaars. Dit maakt die OSS-projecten kwetsbaar, niet alleen vanuit continuïteitsoverwegingen maar ook omdat bij kleinere teams de kans groter is dat er minder volgens professionele standaarden wordt gewerkt en kwetsbaarheden minder snel worden verholpen.

Alvorens een OSS-voorziening te implementeren is het daarmee verstandig een beeld te krijgen van de omvang, samenstelling en werkwijze van de betrokken community en om tijdens het gebruik ook te blijven monitoren hoe de community activiteiten zich ontwikkelen. Het gemiddeld aantal *repo commits* per maand kan een goede graadmeter zijn voor de omvang en activiteit van de community door de tijd heen. Blijf weg bij OSS die al langere tijd niet is geüpdate. De kans is dan groot dat kwetsbaarheden niet tijdig zijn en worden verholpen.

Het gebrek aan SLA's en bijbehorende zekerheden is ook de reden waarom bedrijven en (overheids)organisaties er vaak voor kiezen om OSS alleen in bedrijfskritische processen in te zetten wanneer daaraan een gedegen (betaald) support contract is te koppelen met SLA's. Dit is bijvoorbeeld het geval bij RedHat Linux. Veel OSS kent dergelijke beheercontracten niet. In die gevallen is van belang een goede risicoanalyse uit te voeren en hierbij ook de kwaliteit en robuustheid van de betrokken community in mee te nemen. Bij een robuuste, professionele community kan het dan toch mogelijk zijn om de betreffende OS-voorziening in te zetten voor bepaalde primair proces taken. Een bekend voorbeeld hiervan binnen de cyber security is het open source cyber threat platform [MISP](#).

Ook wanneer een OS-community robuust en professioneel werkt blijft het belangrijk deze te blijven monitoren. De aandacht van de community kan zich gaandeweg verleggen naar andere activiteiten. Dit betekent dat het belangrijk is om, zeker wanneer er geen heldere enterprise beheercontracten kunnen worden afgesloten, altijd de continuïteitsrisico's in ogenschouw te nemen wanneer de keuze voor een OS-voorziening wordt overwogen en daarbij een exit strategie beschikbaar te hebben.

Het ontbreken van een duidelijk eigenaarschap of opdrachtgeverschap bij OSS maakt het ook lastig om *security governance*, dus de besturing van security en security controls, effectief

aan de onderliggende code aanpassingen wil of moet doen, dan is de vraag hoe daarmee het beste kan worden omgaan. Het kan daarbij verstandig zijn om de OSS zodanig intern aan te passen dat een eigen, specifieke variant ontstaat van die OSS. Hierdoor kan de organisatie meer grip en regie houden over de doorontwikkeling en het beheer daarvan, inclusief de beveiligingstechnische aspecten daarbij. Een nadeel is echter dat de OS-community zich vrijwel altijd primair zal richten op de oorspronkelijke OSS-versie, waarvan de nieuwe eigen versie is afgeleid. In het begin betekent dit dat er actief zal moeten worden gevolgd welke aanpassingen er aan die oorspronkelijke versies worden gedaan zodat die ook in de nieuwe versie kunnen worden doorgevoerd. Naarmate de tijd verstrijkt en de versies steeds meer uit elkaar gaan lopen, wordt het verwerken van aanpassingen een grotere opgave en zal de eigen organisatie steeds meer zelf moeten (laten) doen in het beheer en onderhoud van de eigen OSS-versie.

Open source vraagt inzet van SBOM

De voorgenoemde aandachtspunten spelen niet alleen bij het gebruik of bij de ontwikkeling van *open source* maar regelmatig ook bij *closed source software*. Op het moment dat ontwikkelaars eigen software bouwen, bijvoorbeeld in de vorm van maatwerk voor een overheidsorganisatie, dan gebruiken zij hiervoor vaak reeds bestaande, vrijelijk beschikbare software bouwblokken/*libraries*. Zij hoeven die componenten dan niet zelf te bouwen en te testen en dat versnelt het ontwikkelproces. Vaak gaat het in dit soort gevallen om *open source* softwarecomponenten en dat vraagt om aandacht. Op deze manier kan een softwareproduct namelijk afhankelijk worden van een set met OSS-componenten zonder dat de gebruikersorganisatie dit weet. En dat kan vervolgens tot aanzienlijke security risico's leiden, zoals zichtbaar werd bij de recente [Log4J](#) kwetsbaarheid. De OSS-component Log4J bleek toen plots gebruikt te worden in een groot aantal softwareproducten, waaronder

ook in commerciële CSS-producten. De softwareontwikkelaars van die producten hadden de *open source* Log4J gebruikt bij de bouw daarvan, maar dit was bij de gebruikers niet bekend.

Log4J laat zien dat het belangrijk is om bij het ontwikkelen van software, of dit nu OSS of CSS is, altijd vast te leggen welke OSS-componenten daarbij gebruikt zijn. Dit hoeft niet handmatig te gebeuren. Ook hiervoor biedt een geautomatiseerde en goed ingerichte CI/CD-straat een passende oplossing. De CI/CD-straat dient hiervoor te beschikken over technische voorzieningen die in kaart brengen welke (OS) softwarecomponenten gebruikt zijn bij het samenstellen van de code, aan de hand van [Software Composition Analysis \(SCA\)](#). Die in kaart gebrachte componenten en software bouwstenen worden vervolgens vastgelegd in een [Software Bill of Materials \(SBOM\)](#). Op het moment dat (overheids)organisaties zelf OSS ontwikkelen en open aanbieden is het verstandig, ook in het kader van transparantie, om in de OS *repository* steeds de SBOM toe te voegen.

Het hebben van een SBOM/SCA biedt tevens de mogelijkheid om direct en geautomatiseerd de OSS te monitoren op kwetsbaarheden. [Onderzoek](#) van SCA-leverancier Synopsys laat bijvoorbeeld zien dat in 2021 bijna 100% van alle door hen geanalyseerde software, OSS-componenten bevatte en dat ruim 80% van de geanalyseerde software, kwetsbaarheden had in de open source componenten waaruit het mede was opgebouwd.

Daarnaast biedt het nog een ander voordeel. OS-softwarecomponenten kunnen namelijk vaak prima (her)gebruikt worden door organisaties, ook in volledig zelfontwikkelde (CS) software, zolang dit maar wel steeds past binnen de licentievoorwaarden van die OSS componenten. In de praktijk wordt door ontwikkelaars echter lang niet altijd gecontroleerd of de OS-softwarecomponenten die zij toepassen binnen een IT-omgeving of hergebruiken in nieuwe software, ook daadwerkelijk daarvoor

Handreiking veilig gebruik van Open Source

Werk risicogestuurd	Voer een risicoanalyse uit waarbij voor de betreffende OSS-voorziening wordt afgewogen welke risico's gelden en welke beheersmaatregelen nodig zijn, gegeven ook het beoogde inzetgebied van de OSS en de data die daarmee verwerkt zal worden. Weeg in die risicoanalyse ook onderstaande punten mee. Stel de risicoanalyse periodiek bij.
Monitor de OSS community	Evalueer, beoordeel en monitor de bij de OSS betrokken ontwikkel- en beheer community op robuustheid/omvang en professionaliteit. Dit moet borgen dat code goed en veilig wordt ontwikkeld en dat de community groot en duurzaam genoeg is dat beheer ook in de toekomst geborgd is. Ontwikkel en hanteer voor beide aspecten meetbare indicatoren die door de tijd heen gemeten kunnen worden, zoals het aantal git commits per maand.
Monitor release cadans	Wees extra alert op OSS die al meerdere maanden niet is geüpdatet en blijf in principe weg bij OSS die al meer dan een jaar niet is geüpdatet.
Hanteer dezelfde maatstaven als voor eigen code	Weeg, beoordeel en beschouw OSS langs dezelfde lat als die geldt voor intern ontwikkelde code en/of die geldt voor commerciële software. Dat betekent dat OSS aan dezelfde security vereisten, architectuur kaders en code kwaliteitseisen zou moeten voldoen als die intern worden gehanteerd.
Stel heldere eisen aan kwaliteit en security	Stel richting de OSS-community kwaliteits- en security eisen ten aanzien van het ontwikkelproces, zoals het gebruik van een geautomatiseerde CI/CD-straat met software component analyse (SCA), code kwaliteitsbeoordeling en vulnerability scanning voorzieningen. Stel eisen aan code signing en het gebruik van MFA door de ontwikkelaars, en zo mogelijk ook aan de certificering of opleidingen van die ontwikkelaars.
Borg beheer en continuïteit	Bepaal of het beheer en continuïteit van de OSS geborgd zijn, of dat hiervoor additioneel beheercontracten met SLA's kunnen worden afgesloten.
Evalueer ook zelf de code	Evalueer de code kwaliteit en security van OSS in een eigen geautomatiseerde CI/CD-omgeving met behulp van SCA en DAST/SAST voorzieningen.
Hanteer een SBOM	Zorg dat via deze weg ook alle OSS (componenten/libraries ed.) die de organisatie gebruikt, worden opgenomen in een SBOM (Software Bill of Materials) voorziening en gescand worden op kwetsbaarheden. Zonder deze nauw samenhangende technische voorzieningen kan het gebruik van OSS (te) risicovol zijn.
Scan op kwetsbaarheden	Scan permanent op kwetsbaarheden in gebruikte OSS (componenten) en monitor de snelheid waarmee kwetsbaarheden worden gepatcht. Faseer OSS uit waar het patchen bij herhaling te lang blijkt te duren.
Verwerk OSS in interne kaders	Verwerk het gebruik van OSS en de beveiligingstechnische aspecten daarvan in sourcing-, architectuur- en security (governance) kaders. Bepaal bij het in gebruik nemen van OSS hoe de security governance is georganiseerd en of dit aansluit bij vereisten uit geldende wet- en regelgeving.
Verweesde OSS	Het ter beschikking stellen van zelf ontwikkelde software als OSS vraagt dat er tijdig wordt gecommuniceerd wanneer het beheer van die OSS wordt afgebouwd en/of stopt, omdat hierdoor kwetsbaarheden in de software kunnen komen.

Uitgave

Nationaal Cyber Security Centrum (NCSC)
Postbus 117, 2501 CC Den Haag
Turfmarkt 147, 2511 DP Den Haag
070 751 5555

Meer informatie

www.ncsc.nl
info@ncsc.nl
[@ncsc_nl](https://twitter.com/ncsc_nl)

December 2022