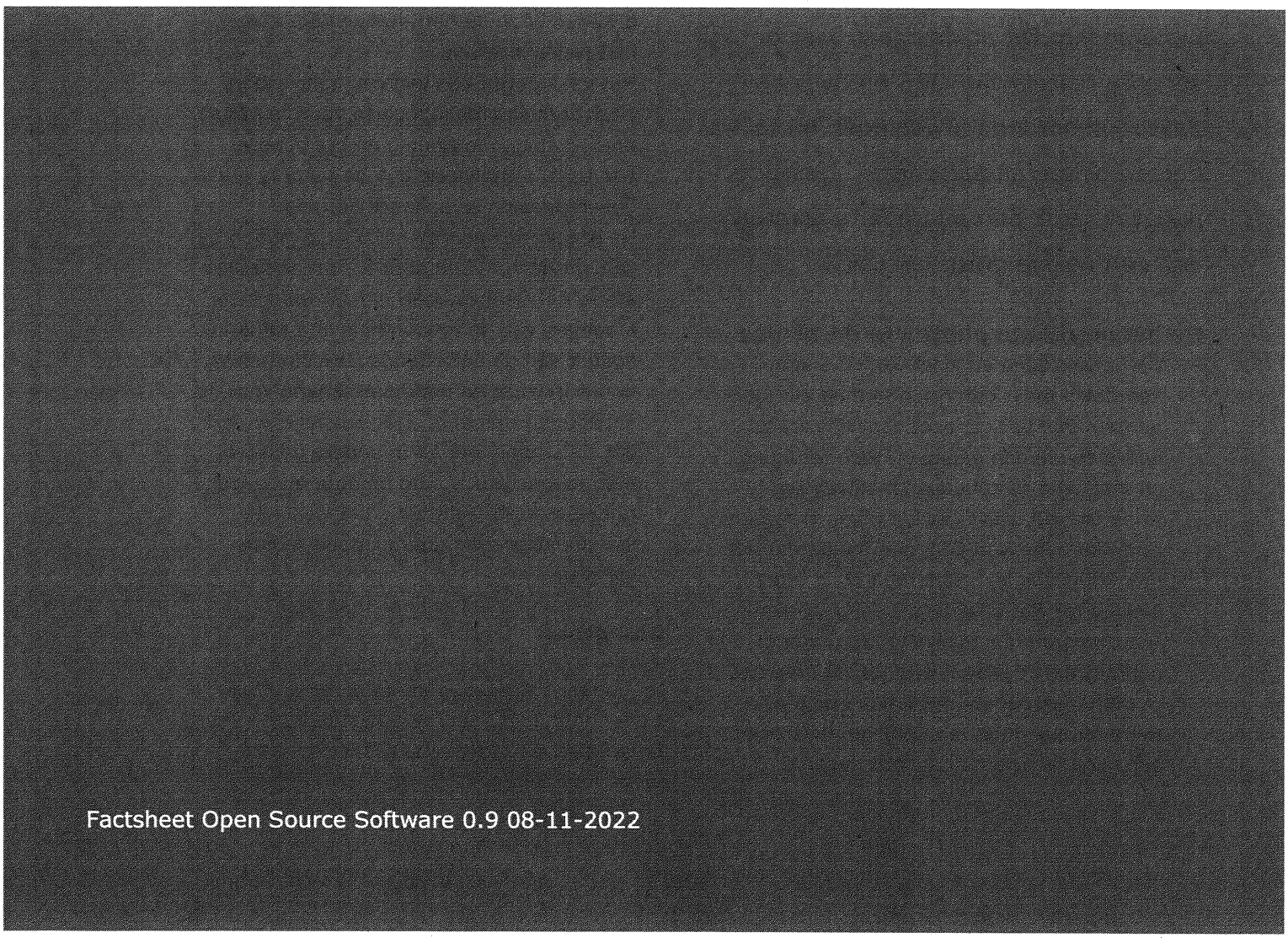
7

Nationaal Cyber Security Centrum
Ministerie van Justitie en Veiligheid

Open Source Security

Een advies voor veilig gebruik van open source sofware



Factsheet Open Source Software 0.9 08-11-2022

Open source software (OSS) heeft de afgelopen decennia wereldwijd een voorname stempel gedrukt op de ontwikkeling van digitale dienstverlening. Op dit moment bestaat het overgrote deel van de digitale producten en diensten zelfs uit één of meerdere OSS-componenten. Open source is daarmee overal.

Het gebruik en de ontwikkeling van OSS kent diverse voordelen en (daardoor) een groeiende belangstelling. Naast voordelen kent de inzet van OSS echter ook specifieke risico's en uitdagingen. Dit advies gaat over de security risico's van OSS en over de daarbij behorende afwegingen en te nemen beheersmaatregelen. Hiermee poogt dit stuk een bijdrage te leveren aan een veilige inzet van OSS.

De belangrijkste punten in dit advies

- De ontwikkeling en inzet van OSS kent voordelen maar ook risico's en aandachtspunten. Een veilige inzet van OSS vraagt daarmee om een gedegen risico afweging en inzet van risicobeheersmaatregelen.
- In de praktijk blijkt dat het open, transparante karakter van OSS niet maakt dat het ook vrij is van kwetsbaarheden, in tegenstelling tot wat vaak wordt aangenomen. Het overgrote deel van de OSS (componenten) die in gebruik zijn, bevat meerdere kwetsbaarheden, soms ook ernstige.
- Het open karakter van OSS kan juist ook bepaalde nieuwe risico's introduceren.
- Veilige OSS vraagt om professionele software ontwikkeling met behulp van moderne, geautomatiseerde ontwikkeltooling zoals een geautomatiseerde CI/CD-straat met daaraan gekoppeld technische voorzieningen voor software composition analysis, application security testing, vulnerability scanning, pentesten en een Software Bills of Materials (SBOM).
- De inzet van OSS kent specifieke continuïteitsrisico's. Er is namelijk veelal niet één specifieke (professionele) partij verantwoordelijk voor de continuïteit, het beheer en het tijdig patchen van software. Dit beheer gebeurt vaak door een collectief van (vrijwillige) ontwikkelaars. Niet zelden is die groep klein en kwetsbaar. Het vraagt van OSS-gebruikers dat zij de ontwikkeling, professionaliteit en robuustheid van de betreffende OSS-community continu blijven volgen. En op basis hiervan blijven afwegen of zij de betreffende OSS al dan niet willen inzetten.
- Het vaak ontbreken van een aanwijsbare eindverantwoordelijke partij voor de (door) ontwikkeling, beveiliging en het beheer van OSS, compliceert ook de security governance en compliance borging.
- De specifieke security risico's die gelden bij OSS vragen om een goede, continue risico afweging. De inzetbaarheid zal ondermeer afhangen van de robuustheid en professionaliteit van de betreffende OS-community, hun werkwijze en het inzetscenario voor de betreffende software. Dat kan betekenen dat OSS soms wel, en in andere gevallen niet, toepasbaar is. Om bij deze afweging te helpen staat achterin dit advies een 'handreiking veilig gebruik open source software'.

Doelgroep

Dit advies is bedoeld voor diegenen die betrokken zijn bij IT sourcingsbeslissingen zoals CIO's, IT managers, CISO's, IM adviseurs en architecten.

Aanleiding voor dit advies

Open source software (OSS) heeft de afgelopen decennia wereldwijd een belangrijke stempel 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 ander 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 onderzoeken[1] 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.

Het feit dat de code van OSS voor eenieder inzichtelijk is, leidt in de praktijk dus niet noodzakelijkerwijs tot minder kwetsbaarheden. Mogelijk heeft dit ermee te maken dat het open en transparant maken van code nog niet automatisch betekent dat die code ook daadwerkelijk door anderen systematisch wordt gereviewd en verbeterd. Een andere reden, die verderop nader wordt toegelicht, heeft ermee te maken dat OSS die wordt (her)gebruikt soms al voor langere tijd niet meer is bijgewerkt en geüpdate. In de tussentijd bekend geworden kwetsbaarheden zijn dan niet gepatcht en zijn blijven bestaan. Uit recent onderzoek blijkt dat 88% van de geanalyseerde OSS, onderdelen bevatte die 2 jaar of langer daarvoor voor het laatst waren geüpdate.

Het open en transparante karakter van OSS brengt daarnaast zelf overigens ook een aantal risico's met zich mee. Het maakt het voor kwaadwillenden bijvoorbeeld eenvoudiger om zelf (nog onbekende) kwetsbaarheden in de OSS te vinden en te misbruiken. In sommige software worden – hoewel dit vanuit security technisch oogpunt eigenlijk niet hoort – ook zogenoemde *hardcoded credentials* gebruikt. Feitelijk gaat het hierbij om wachtwoorden die zijn verwerkt in de software code en die nodig zijn om de betreffende software bijvoorbeeld toegang te laten krijgen tot andere applicaties of tot data bronnen (API's). In OSS zijn die hardcoded credentials, vanwege het open karakter, eenvoudig terug vindbaar.

De (vaak collectieve) wijze waarop OSS wordt ontwikkeld en gedistribueerd kan het ook eenvoudiger maken voor kwaadwillenden om gericht en al dan niet verhuld nieuwe kwetsbaarheden of malware toe te voegen aan de OSS-code. In het geval van CSS is de productie- en distributieketen van software meer afgeschermd en afgesloten voor anderen dan het kern ontwikkelteam en doet dit risico zich minder voor. Het recente voorbeeld van node.ipc laat ook zien dat het gericht toevoegen van kwaadaardige code aan OSS geen theoretisch risico is, maar één met in potentie verstrekkende gevolgen. Andere voorbeelden zijn die van Linux Mint (2016) en Linux Gentoo (2018). De problematiek rondom de Solarwinds hack laat zien dat CSS hier overigens ook niet helemaal immuun voor is. Het risico dat kwaadwillende misbruik maken van het open karakter van OSS door zelf malware toe te voegen aan de source code is in zekere mate te beheersen door eisen te stellen aan *code signing* en het gebruik van multifactor authenticatie (MFA) door ontwikkelaars.

Transparantie kan daarnaast nog een ander meer algemeen risico met zich mee brengen wanneer de open source code betrekking heeft op technische voorzieningen voor defensie doeleinden, vitale infrastructuur of nationale veiligheid. Vreemde mogendheden kunnen dan via de OSS-code direct inzicht verkrijgen in technische capabilities en technische (on)mogelijkheden van (overheids)organisaties die in deze domeinen actief zijn. Dit kan betekenen dat inzet van OSS hier onwenselijk is of in ieder geval inzet van specifieke aanvullende maatregelen vereist.

Het vraagstuk van continuïteit, support en *security governance*

Eén van de belangrijkste belemmerende factoren in het gebruik en de adoptie van OSS binnen de private en publieke sector heeft te maken met de continuïteitsrisico's die samenhangen met OSS. OSS wordt over het algemeen ontwikkeld en beheerd door een community van vrijwilligers, geïnteresseerden en gebruikers van de betreffende software. Het kan zijn dat gedurende een bepaalde periode een specifieke partij, zoals een overheidsorganisatie of bedrijf, de ontwikkeling en het beheer ervan (co)financiert of regisseert, maar ook dan zal vaak het streven zijn om de ontwikkelde OSS uiteindelijk te laten landen in een brede community die voor de verdere doorontwikkeling en het beheer zorg draagt. De community van bouwers en beheerders biedt die OSS gratis, en veelal in open source licentie,

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

te organiseren. Want wie is er uiteindelijk verantwoordelijk voor de beveiliging van de OSS? Wie stelt de kaders of wie bepaalt de beveiligingseisen of security architectuur? Als die afspraken en vereisten er al zijn, wie implementeert, controleert, valideert en borgt die dan? En wie is daar dan uiteindelijk ook weer op aanspreekbaar? Vaak is dit onduidelijk of in ieder geval omkleed met onzekerheden, en dat brengt security risico's voor de gebruikers van die OSS met zich mee. Dit aspect zullen partijen die OSS willen gebruiken dan ook mee moeten wegen in hun risicoafweging. Logischerwijs heeft OSS, vanwege het andere risico's die er spelen bijvoorbeeld op dit vlak, ook een aparte plek in de security architectuur en security governance kaders van een organisatie.

Open source veilig ontwikkelen en integreren met een CI/CD-straat

Op het moment dat onder regie van een (overheids)organisatie of voor een (overheids)organisatie OSS wordt ontwikkeld, is het van belang dat hierbij dezelfde standaarden, eisen en *best practices* worden gehanteerd, als die gelden voor iedere vorm van professionele softwareontwikkeling. De kwaliteit en veiligheid van de software hangt zoals hiervoor al aangegeven im-mers vooral ook af van de professionaliteit waarmee deze wordt ontwikkeld. Concreet betekent dit onder meer dat OSS-ontwikkeling dient te verlopen via een geautomatiseerd CI/CD-proces waarbinnen diverse geautomatiseerde checks, controles en validaties plaats vinden, in het bijzonder ook op het gebied van security. Het gaat hierbij dan onder andere om geautomatiseerde *Static-, Interactive- en Dynamic Application Security Testing* (respectievelijk SAST, IAST en DAST), *vulnerability scanning* en geautomatiseerd *pentesten*. Op het moment dat de een (overheids)organisatie OSS laat (door)ontwikkelen, is het daarmee belangrijk dat deze aan de ontwikkel community dergelijke eisen meegeeft.

Het hanteren van een dergelijk CI/CD-proces past bij de bredere opgave om te komen tot veilige software ontwikkeling, de implementatie van *security by design* en de zogenoemde '*shift left*' in beveiliging. Het valideren en controleren van OSS in een CI/CD omgeving is niet alleen van belang voor het zelf ontwikkelen van OSS maar ook relevant in het geval een organisatie besluit om elders ontwikkelde, en dus reeds bestaande OSS, te gaan gebruiken. Het voordeel van OSS is namelijk dat het 'open' is en daarmee ook geanalyseerd en gecontroleerd kan worden in de CI/CD-omgeving van de gebruikersorganisatie. De uitkomsten van die geautomatiseerde controles, checks en validaties borgen niet alleen dat er veilige software wordt ingezet (Zitten er bijvoorbeeld kwetsbaarheden in de software?). Ze geven ook inzicht in de kwaliteit en daarmee ook professionaliteit en robuustheid van het OSS-ontwikkeltraject en betreffende OSS community. Het is daarmee ook aan te bevelen om bij een implementatie, de betreffende OSS altijd eerst door te lichten in de eigen CI/CD omgeving als een soort technische *due diligence* evaluatie.

Een organisatie die OSS wil gebruiken of laten ontwikkelen kan behalve deze achteraf controleren en valideren, zo mogelijk ook vooraf eisen stellen aan de ontwikkelmethode, de te gebruiken technologie/stack en de inzet van een geautomatiseerde CI/CD-straat. Evenzo kunnen ook eisen gesteld worden aan de kennis en ervaring van de betrokken ontwikkelaars, zoals dat zij bepaalde open source opleidingen hebben gevolgd of over bepaalde (OS) techniek certificeringen beschikken.

Forking, cloning & branching

Professioneel werken met OSS omvat naast het hanteren van een goed uit-geruste CI/CD-straat ook het werken op basis van het '*build from source*' principe en het adequaat hanteren van *forking, cloning* en *branching*.

Op het moment dat een organisatie OSS-voorzieningen op een zodanige manier wil gaan inzetten binnen de eigen IT-omgeving dat het

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

kunnen worden ingezet. Dit kan leiden tot juridische issues en compliance risico's. Om die reden is het van belang om de SCA te koppelen aan licentie- en compliance analyse, zodat bij het (geautomatiseerd) samenstellen van de SBOM ook de relevante (open source) licenties daarin kunnen worden vastgelegd. Volgens SCA leverancier Synopsys blijkt dat in 2021 bij ruim 50% van de de door hen geanalyseerde software sprake was van open source licentie issues. Dit is daarmee een niet te veronachtzamen aandachtspunt.

Software Bill of Materials

Wil je meer weten over de inzet van de SBOM in het CI/CD-proces en waarom dit van belang is vanuit cyber security perspectief, lees dan het rapport van het onderzoek dat het NCSC samen met CapGemini hiernaar heeft gedaan.

Helderheid over 'verweesde' OSS

Een (overheids)organisatie die zelf OSS heeft ontwikkeld of laten ontwikkelen en dit aan eenieder voor hergebruik aanbiedt onder open source licentie, zal op een gegeven moment voor het besluit staan om de doorontwikkeling en uiteindelijk ook het beheer ervan af te schalen dan wel geheel stop te zetten. De organisatie stopt er dan geen tijd of geld meer in, maar anderen kunnen de OSS eventueel wel nog gebruiken. Dit moment van afschalen en stopzetten vraagt om de nodige aandacht bij de (overheids)organisatie die langere tijd als sponsor en/of eigenaar optrad. Op het moment dat de aandacht voor beheer afneemt of zelfs helemaal stopt, groeit de kans dat kwetsbaarheden in de software niet meer tijdig worden opgemerkt en hersteld. Er beginnen 'gaten' in de OSS te vallen. En die kwetsbaarheden en security risico's kunnen negatief afstralen op de (overheids)organisatie die lange tijd zo verbonden was met die software. Dit betekent dat het belangrijk is dat sponsoren of eigenaren van OSS tijdig en expliciet aangeven dat zij het beheer en de doorontwikkeling van die OSS afschalen en uiteindelijk stopzetten, én dat daardoor kwetsbaarheden in de (verweesde) software kunnen gaan ontstaan die niet tijdig verholpen zullen worden. Gebruikers van die OSS kunnen hiermee dan weloverwogen hun eigen afwegingen maken

Tot slot

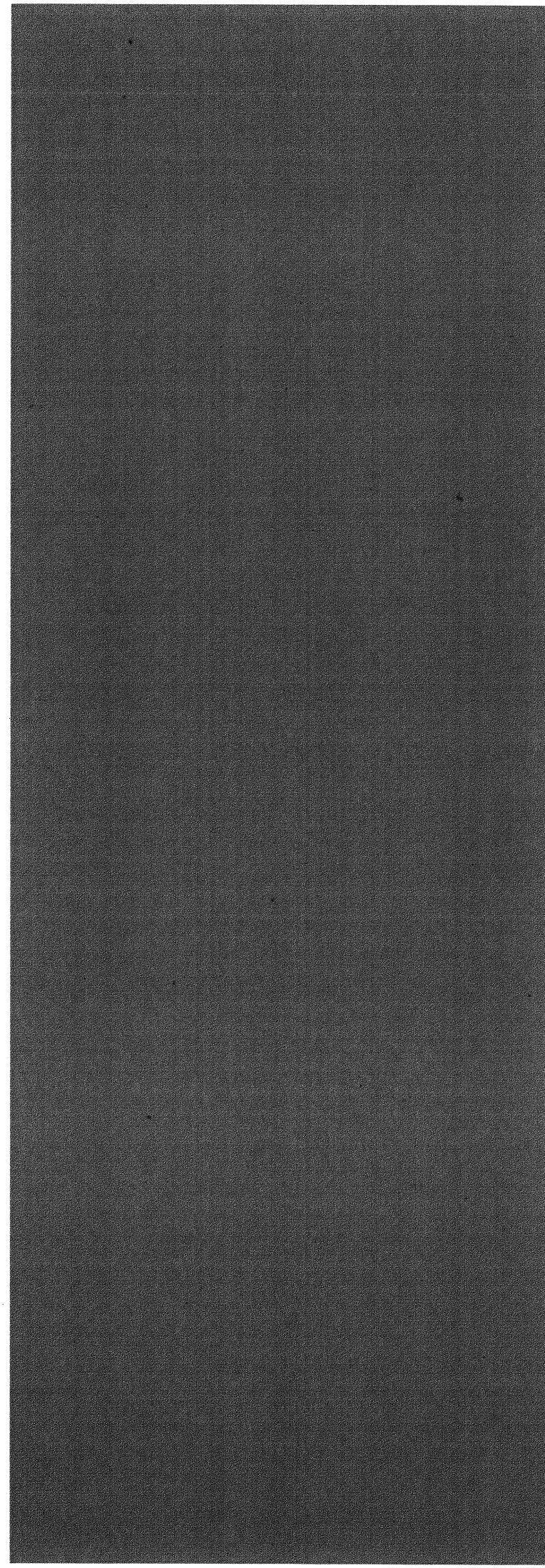
Of het inzetten dan wel zelf (laten) ontwikkelen van OSS voor (overheids)organisaties een verstandige, veilige keuze is, hangt af van de context en condities waarbinnen dit plaats moet vinden. Dit vraagt steeds om een adequate risicoanalyse en sourcingsafweging, en die kan goed per OSS-oplossing en per inzetscenario verschillen. Die afweging kan ook veranderen gedurende de *life cycle* van de betreffende OSS, bijvoorbeeld wanneer de werkwijze van de OSS community verandert.

In algemene zin geldt dat inzet van OSS op business kritische processen veelal om andere afwegingen en andere risicobeheersmaatregelen zal vragen dan wanneer er sprake is van inzet op secundaire procesgebieden.

OSS-ontwikkeling en beheer kent een andere dynamiek en daarmee ook andere uitdagingen en risico's dan die te vinden zijn bij CSS- of COTS-oplossingen. Dit geldt voor de gehele *service life cycle*. Het vereist kennis en ervaring met die OSS-specifieke risico's en uitdagingen en bijvoorbeeld ook het besef dat gedurende de life cycle enige vorm van voeling en contact met de betrokken OSS-community nodig is, temeer om ook de risicoblootstelling goed te kunnen blijven volgen.

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 continuiteit	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

[maand] [jaar]