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. Dat kan betekenen dat de OS-voorziening niet geschikt is voor bepaalde bedrijfskritische processen en bijvoorbeeld alleen kan worden ingezet in niet-kritieke innovatie- of test omgevingen.
- 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 immers 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.
- 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 kwa-

liteit 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. Een voorbeeld hiervan is de OpenSFF 'developing secure software' certificering.

Forking, cloning & branching
- Professioneel werken met OSS omvat naast het hanteren van een goed uitgeruste 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.

Risico's van OS vragen ook om aandacht bij de ontwikkeling en inzet van CSS: de noodzaak van een 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, die jaarlijks SCA statistieken publiceert, 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.

  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

Expliciet communiceren over ontstaan van 'verweesde' OSS

- Een (overheids)organisatie die zelf OSS heeft ontwikkeld of laten ontwikkelen en dit aan eenieder voor hergebruik aanbiedt, zal op een gegeven moment voor het besluit staan om de doorontwikkeling en uiteindelijk ook het beheer ervan af te schalen dan wel 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. Inzet op business kritische processen vraagt om andere afwegingen en andere risicobeheersmaatregelen dan wanneer er sprake is van secundaire procesgebieden.
- OSS-ontwikkeling en beheer kent een andere dynamiek en daarmee ook andere uitdagingen en risico's dan die te vinden zijn 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 Software

- De inzet en ontwikkeling van open source software (OSS) heeft verschillende voordelen, maar kent ook diverse risico's en uitdagingen. Alvorens OSS op een risicoarme en effectieve manier in te kunnen zetten, dienen de beveiligingsrisico's op een adequate manier beheerst te (kunnen) worden. Hoe dat kan staat in de punten hieronder op hoofdlijnen uitgewerkt. Daar waar het gaat om de inzet van OSS geldt op die manier een **'Ja, mits'** benadering.

- **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.

- **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.

- **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.

- **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 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.

- **Bepaal of het beheer en continuïteit van de OSS geborgd zijn**, of dat hiervoor additioneel beheercontracten met SLA's kunnen worden afgesloten.

- **Evalueer de code kwaliteit en security van OSS in een eigen geautomatiseerde CI/CD-omgeving** met behulp van SCA en DAST/SAST voorzieningen.

- **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 zo 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 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.

- Het ter beschikking stellen van zelf ontwikkelde software als OSS kent verschillende uitdagingen en (afbreuk)risico's. Borg in dat kader dat er tijdig wordt gecommuniceerd wanneer het beheer van die OSS wordt afgebouwd en/of stopt, omdat hierdoor kwetsbaarheden in de software kunnen komen.