Nationaal Cyber Security Centrum
Ministerie van Justitie en Veiligheid

Open Source Security

Software Supply Chain Security en Open Source

Factsheet Open Source Software | Versie 2.1 | 24-11-2023

Open Source Software (OSS) is software waarvan de broncode openbaar beschikbaar is en in sommige gevallen verspreid mag worden. 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 Open Source componenten.

Software Supply Chain Security (SSCS) staat voor de werkwijze die wordt gebruikt om beveiligingsrisico's, bij de ontwikkeling, distributie en implementatie van software, tot een minimum te beperken. SSCS is een bouwsteen voor het uitvoeren van een risicobeoordeling en het bepalen van risicobeheersingsmaatregelen. Daarbij worden maatregelen voorgesteld om de betrouwbaarheid van softwarecode te optimaliseren en ongeautoriseerde wijzigingen in het ontwikkel- en uitleverproces te voorkomen.

Hoewel SSCS een generieke methode is, die moet worden toegepast ongeacht de licentievorm of openheid van broncode, zijn er een aantal aspecten die bij het gebruik van Open Source specifiek de aandacht verdienen. Daar gaat deze factsheet verder op in.

Leeswijzer

Ten behoeve van de leesbaarheid worden eerste de losse onderwerpen en hun relevantie toegelicht, waarna een verband tussen de verschillende aspecten wordt besproken. De volgende invalshoeken worden besproken:

- Software Supply Chain Security in het kort: wat houdt SSCS in?
- Open Source: een korte inleiding van Open Source en de relevantie ervan voor de doelgroepen van het NCSC.
- Open Source en Software Supply Chain Security: het verband tussen de twee onderwerpen.
- Aandachtspunten rondom de Software Supply Chain bij het gebruik van Open Source Software: hier leest u de belangrijkste bevindingen en adviezen van NCSC rondom beide onderwerpen.
- Open Source en Wet Open Overheid: een uitstap naar aspecten van SSCS vanuit perspectief van de vrijgever van de broncode.

Doelgroep

Dit advies is bedoeld voor CIO's, IT-managers, CISO's, IM-adviseurs en -architecten.

Samenwerkingspartners die hebben bijgedragen aan de inhoud

Ministerie van Binnenlandse Zaken (AIVD/NBV, Beleid Open Source en Logius), het Ministerie van Volksgezondheid, Welzijn en Sport (Concern CISO Office), SURF.

Software Supply Chain Security

SSCS als begrip is breed toepasbaar en staat los van de softwarelicentievorm of beschikbaarheid van de broncode. Op hoofdlijnen bestaat het uit de volgende aspecten.[1]

Tijdens de ontwikkeling van software wordt zo veel mogelijk voorkomen dat broncode fouten bevat die kwetsbaar zijn voor aanvallers. Code die wordt uitgeleverd aan afnemers moet daadwerkelijk door de softwareontwikkelaars zijn gepubliceerd. Bij de uitlevering en implementatie moet door de afnemer de **integriteit van de software worden gevalideerd.** Ook wordt gevalideerd dat de software volgens beveiligingsrichtlijnen is geconfigureerd en welke andere softwarecomponenten worden aangeroepen.

Sub-componenten, zoals (externe) *libraries* en modules die noodzakelijk zijn voor het draaien van de software, worden op hun beurt **geïnventariseerd en aan een vergelijkbaar proces onderworpen.**

Deze uitgangspunten worden op verschillende manieren toegepast en zijn afhankelijk van de te beschermen informatie, noodzaak van (hoge) beschikbaarheid van informatie en de omgevingsanalyse van de organisatie. De afnemer past de software toe als onderdeel van haar digitale dienst en toets deze naar aanleiding van haar risico-analyse: **is de (digitale) dienst, inclusief álle daarvoor gebruikte software, veilig genoeg binnen de risico-context van onze organisatie?** Dit wordt afgemeten op basis van een door de organisatie vastgesteld control framework, waar genoemde aspecten van SSCS nadrukkelijk onderdeel van zijn.

Deze sterk versimpelde weergave van SSCS is van toepassing op álle software die een organisatie gebruikt: OSS én Closed Source Software (CSS).

Open Source in het kort

Het internet is gebaseerd op open standaarden en in belangrijke mate gefundeerd op Open Source Software. Daarnaast neemt de belangstelling en toepassing van Open Source licentie- en werkvormen nog steeds toe. Dit geldt ook voor de overheid, die zich tot doel heeft gesteld om zelf ontwikkelde software als Open Source vrij te geven via het 'open, tenzij'[2] principe.

Onderzoekers zien dat er in de praktijk weinig verschil zit in het aantal en soort kwetsbaarheden tussen OSS en CSS[3]. In sommige gevallen wordt specifiek voor een Open Source vorm gekozen als beheersmaatregel vanuit de risico-analyse, bijvoorbeeld omdat veelgebruikte Open Source encryptiecomponenten het beste aan de gestelde voorwaarden voldoen. Daarbij worden soms aanvullende maatregelen genomen zodat door de afnemer gedefinieerde SSCS controls passend worden ingevuld.

Zo werd de veelgebruikte software OpenVPN voorzien van het compacte PolarSSL, in opdracht van verschillende departementen van de rijksoverheid. De combinatie van beide Open Source projecten maakte het mogelijk om de code in zijn geheel onafhankelijk te auditen, aangezien PolarSSL veel minder code

[1] [2209.04006] What is Software Supply Chain Security? (arxiv.org)

[2] Staatssecretaris Knops: 'Geef broncode van overheidssoftware waar mogelijk vrij' - Digitale Overheid

[3] 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.

bevat dan OpenSSL – wat standaard met OpenVPN wordt meegeleverd.

Daarnaast is de code 'evaluatie-klaar' gemaakt en gehardend door onveilige opties te elimineren. Voorts is de Software Supply Chain passend gemaakt bij vereisten: er zijn contractuele afspraken gemaakt met een derde leverancier zodat kwetsbaarheden worden gesignaleerd, geduid en zo nodig aangepakt. Door deze benadering van een op Open Source gebaseerd encryptieproduct kon OpenVPN-NL[4] onder VIR-BI worden geaccrediteerd en goedgekeurd voor gerubriceerde informatie tot niveau Departementaal-Vertrouwelijk.

Introductie: Open Source Software en Software Supply Chain Security

Overheden en organisaties gebruiken tal van Open Source softwareoplossingen en -componenten in hun huidige en toekomstige (digitale) diensten. Organisaties en overheden zoeken naar concrete handvatten bij het beoordelen van de verschillende aspecten van Open Source in de context van hun risico-beoordeling[5], zowel in de rol van afnemer als in de rol van softwareproducent.

Vanuit cybersecurity perspectief wordt in risicobeoordelingen door organisaties regelmatig vastgesteld dat het onbedoeld toevoegen van code aan een nieuwe release van een softwareproduct, bedoeld voor kwaadaardige doeleinden, een realistisch risico vormt. Daarnaast wordt vaak beoordeeld dat grip op alle gebruikte softwarecomponenten, of juist het gebrek daaraan, ook een realistisch risico vormt waar beheersmaatregelen voor moeten worden gedefinieerd. Daarom zoeken producenten en gebruikers naar passende risicobeheersingsmaatregelen. Voorkomen dat onbedoeld kwaadaardige code, of een kwetsbaar softwarecomponent, worden gebruikt is de essentie van Software Supply Chain Security.

[4] OpenVPN-NL (fox-it.com)

[5] Governments view open source as critical for enhancing digital services, experts say - GCN

Aandachtspunten rondom de Software Supply Chain bij het gebruik van Open Source Software

De belangrijkste feiten

Een veilige inzet van zowel Open als Closed Source Software vraagt om een gedegen risico-afweging en inzet van risicobeheersmaatregelen.[6]

Veiligheid van software wordt bepaald door de kwaliteit van productie- en publicatieprocessen, ongeacht de openbaarheid of geslotenheid van broncode.

Controle over de Software Supply Chain verdient aandacht bij het gebruik van OSS.

Inzicht in de Software Bill of Materials, met daarop de (her)gebruikte externe OSS en CSS componenten, is relevant voor het invullen van een Software Security control framework.

Open broncode is geen indicator van code-kwaliteit

Het openstellen van broncode of het gebruiken van openbare broncode geeft op zichzelf geen voorspelbare veiligheidsuitkomsten. Openbaarheid is geen indicator van kwaliteit van softwarecode.[7]

De kwaliteit van achterliggende productieprocessen en de wijze waarop software in een productieproces wordt beheerd zijn van grotere betekenis. OSS projecten worden op verschillende manieren ingevuld. Zo zijn er grote, veelgebruikte Open Source producten waarvan de ontwikkeling wordt ondersteund door één of meerdere grote bedrijven, of waarbij zowel ontwikkelcapaciteit als volledige ontwikkelprocessen worden gesubsidieerd door commerciële bedrijven én (non-)gouvernmentele organisaties.

Ook zijn er veelgebruikte Open Source oplossingen die worden ontwikkeld door vrijwillige OSS-communities met een volwassen productieproces, transparant feedbackproces, regelmatige, actieve code-reviews en de mogelijkheid om patches van derden te kunnen verwerken in het eindproduct.

Er zijn ook projecten die worden onderhouden door slechts één of enkele ontwikkelaars. Het aantal ontwikkelaars is wederom geen indicator van de code-kwaliteit, maar kan wijzen op beperkte capaciteit of reactiesnelheid in het geval van een beveiligingsprobleem. Iedereen kan Open Source publiceren of onder bepaalde voorwaarden bijdragen aan bestaande Open Source projecten.

Daarom is het van belang om bij het gebruik van OSS kennis te nemen van de kwaliteit van achterliggende ontwikkelprocessen. Daar kan rekenschap aan worden gegeven bij het vaststellen van een risico-afweging, inzet van beheersmaatregelen en het vaststellen van security controls.

Inzicht in de Software Supply Chain is van groot belang

De wijze waarop OSS wordt ontwikkeld en gedistribueerd kan het in bepaalde gevallen eenvoudiger maken voor kwaadwillenden om gericht, al dan niet verhuld, nieuwe kwetsbaar-heden of malware toe te voegen aan de OSS-code. Hoogvolwassen OSS-communities hanteren quality-assurance procedures alvorens code te accepteren van derden. In volwassen OSS-communities is het gebruikelijk

[6] Factsheet Risico's beheersen: de waarde van informatie als uitgangspunt. | Factsheet | Nationaal Cyber Security Centrum (ncsc.nl)

[7] Fagundez et al, 2018, p.16

om door middel van een autorisatiemodel te werken. Hoogvolwassen OSS-communities hanteren quality-assurance procedures alvorens code te accepteren van derden. Daarin mogen de aandragers van code de integratie naar het eindproduct niet zelf accorderen. Ontwikkelaars moeten in dergelijke communities eerst bewijzen dat zij voldoende kennis van het product hebben, alvorens in aanmerking te komen voor meer rechten.

Wanneer een community gedreven OSS-ontwikkelproces op een dergelijke manier wordt ingeregeld wordt gewerkt op basis van een prestatiegedreven procedure. Dit zorgt voor een vertrouwensband tussen ontwikkelaars. Alleen hoogwaardige, sterk ontwikkelde Open Source communities beschikken over een dergelijk niveau van kwaliteitsborging. **Het is daarom belangrijk om als afnemer een goed beeld en afdoende garanties te hebben voor de wijze van kwaliteitsborging.**

Het organiseren van afdoende inzicht in de kwaliteitsborging kan, afhankelijk van de zwaarte van de te mitigeren risico's, een forse investering met zich meebrengen of in de praktijk lastig of niet zijn te organiseren. Als inzicht niet, of niet in voldoende mate mogelijk is dan vormt dat een extra risico dat de gebruikersorganisatie mee moet wegen.

Kwaliteitsborging en afspraken over kwaliteit van de broncode is ook in contractueel opzicht relevant.Deze verantwoordelijkheid moet duidelijk worden belegd en in overeenkomsten in ieder geval niet expliciet worden uitgesloten, zoals dat bij sommige CSS-overeenkomsten het geval kan zijn[8].

Software repositories zijn een aantrekkelijk doelwit

Het recente voorbeeld van node.ipc[9] laat zien dat het gericht toevoegen van kwaadaardige code aan OSS een realistisch risico is. Andere voorbeelden zijn die van Linux Mint[10] (2016) en Linux Gentoo[11] (2018).

In het geval van CSS is de productie- en distributieketen van software meer afgeschermd en afgesloten voor anderen dan het kern ontwikkelteam. De problematiek rondom de Solarwinds hack[12] en meer recent de aanval via de 3CX-supply chain[13] laat echter zien dat CSS niet immuun is voor supplychain aanvallen. De supplychain is sinds 2018 een belangrijke *vector* voor het distributie-aspect van complexe cyberaanvallen, in het bijzonder wanneer deze worden uitgevoerd door statelijke actoren.[14]

Het beoordelen van de integriteit van de Software Supply Chain in het Software Security proces is dus belangrijk, evenals het (laten) stellen van de juiste garanties.

Waar bij CSS vaak in contractuele constructies zekerheden op dit onderwerp worden afgedwongen, **vereist OSS in sommige**

[8] CISA: Hold software makers liable for selling insecure tech • The Register

[9] A Developer Altered Open Source Software to Wipe Files in Russia | WIRED

[10] Beware of hacked ISOs if you downloaded Linux Mint on February 20th! – The Linux Mint Blog

[11] Github Gentoo organization hacked - resolved – Gentoo Linux

[12] A Year After the SolarWinds Hack, Supply Chain Threats Still Loom | WIRED

[13] Update en handelingsperspectief supplychain-aanval 3CX | Nieuwsbericht | Nationaal Cyber Security Centrum (ncsc.nl)

[14] Breaking Trust – Shades of Crisis Across an Insecure Software Supply Chain | USENIX

gevallen een actieve benadering en toetsing van het publicatieproces zelf, of het betrekken van een deskundige marktpartij die dergelijke supplychain zekerheden kan aanvullen. Naast de genoemde OpenVPN-NL oplossing zijn tal van wereldwijde voorbeelden beschikbaar: Nginx, 's werelds meest gebruikte (Open Source) web engine[15], kan bijvoorbeeld worden afgenomen in een vorm met garanties en professionele ondersteuning.

Inzicht in álle gebruikte software, ook de sub-componenten, is essentieel voor OSS én CSS

Digitale omgevingen bestaan uit een opeenstapeling van software-afhankelijkheden die modulair zijn opgebouwd. Inzicht in alle gebruikte modules is een essentieel onderdeel van Software Supply Chain Security.

Softwareontwikkelaars gebruiken bestaande software bouwblokken in de vorm van zogenaamde *libraries* of *modules*. Zij hoeven die componenten dan niet zelf te bouwen en te testen, wat het ontwikkelproces versnelt. Het gebruik van bestaande (OSS) componenten kan worden ingegeven vanuit perspectief van beveiliging. Het wordt ontwikkelaars bijvoorbeeld sterk afgeraden om hun eigen encryptiemodule te ontwikkelen, voor hun specifieke toepassing. Daarvoor kan het beste een bestaande library worden gebruikt die uitvoerig is onderzocht, waarvan de werking algemeen bekend is, zich in de praktijk heeft bewezen en wordt ondersteund door een wereldwijd netwerk van specialisten en belanghebbenden.

Het inbouwen van afhankelijkheid van externe *libraries* impliceert een extra verantwoordelijkheid die moet worden ingevuld gedurende de levenscyclus van het softwareproduct. De ontwikkelaar moet vanaf dat moment bijhouden of de software gebruik maakt van veilige, recente *libraries*. Afnemers moeten worden geïnformeerd over belangrijke informatie omtrent deze afhankelijkheden. Software moet snel worden aangepast om gebruik te maken van nieuwere *libraries* in geval van een beveiligingsprobleem.

Het inzicht in software-afhankelijkheden is ook een verantwoordelijkheid voor de afnemer. Die moet zich ervan verzekeren dat (informatie over) alle gebruikte componenten voldoen aan het control framework. Bij voorkeur wordt deze verantwoordelijkheid in samenspraak met de softwareleverancier of software-community ingevuld. Daarbij wordt, in een ideale situatie, een Software Bill of Materials (SBOM) opgeleverd[16] als onderdeel van elke softwaredistributie. Deze wordt vervolgens gemonitord en als input voor risico- en scenariobeoordeling gebruikt.

Het is belangrijk om uw volwassenheidsniveau vast te stellen en de verwachtingen omtrent implementatie van SBOM, of een vergelijkbaar inzicht, op waarde te schatten. Het is in dit stadium soms ingewikkeld of zelfs niet mogelijk om het volledige overzicht te verkrijgen. Dit kan een mate van schijnveiligheid creëeren, wanneer niet op het juiste niveau ingestoken en uitgevoerd. Stel daarom realistische verwachtingen, die passen bij uw volwassenheidsniveau, risicoprofiel en de mate waarin u en uw softwareleveranciers over complete informatie kunnen beschikken.

Houd daarnaast rekening met de verschillende vormen waarin (sub)componenten worden gedistribueerd. Voor een container-architectuur is het bijvoorbeeld mogelijk dat software via een andere route wordt gedistribueerd dan vanuit een bekende, centrale repository waarin deze in eerste instantie is gepubliceerd. Kies bewust voor

[15] Web Server Survey | Netcraft News

[16] Software Bill of Materials (SBoM) en cybersecurity | Onderzoek | Nationaal Cyber Security Centrum (ncsc.nl)

distributiekanalen die u gebruikt voor het verversen van containers.

Ditzelfde geldt voor hardware firmware. Er zijn veel devices, bijvoorbeeld wifi-accesspoints of netwerkswitches, waarvan de firmware gedeeltelijk of volledig is gebaseerd op opensource software. Van elk softwarecomponent moet de authenticiteit van de gebruikte code en componenten herleidbaar zijn. De software moet worden aangeleverd op een manier die past bij uw beveiligingsvereisten.

Een start maken met het opstellen van een Software Bill of Materials (SBOM)

TNO heeft in opdracht van het NCSC een Startersgids opgesteld voor het uitvoeren van Software Composition Analysis (SCA) en het vastleggen van een Software Bills of Materials (SBOM) binnen organisaties met een hoog volwassenheidsniveau en een focus op vulnerability management.[17] Hierin vindt u handvatten voor het ontwerpen en beheren van een SBOM.

Inzicht in gebruikte softwarecomponenten is ook bij het gebruik van CSS relevant. Vaak worden verschillende Open Source componenten meegeleverd in Closed Source software distributies. Uiteraard worden regelmatig ook Closed Source componenten van derde partijen als onderdeel van distributies meegestuurd. Het is in dit kader wederom van belang om vast te stellen of er afdoende informatie, bij voorkeur in de vorm van een SBOM, beschikbaar is om aan de vastgestelde security controls te voldoen.

Een gebrek aan inzicht in de samenstelling van de software leidt tot security incidenten, zoals zichtbaar werd bij de Log4J kwetsbaarheid. Een grote hoeveelheid Open- en Closed Source softwareproducten[18] gebruikten een populair *Open Source Java logging framework* welke kwetsbaar bleek voor *arbitrary code execution*. Door gebrek aan inzicht in gebruikte softwarecomponenten waren veel gebruikers niet in staat om binnen de door henzelf gestelde termijnen te reageren op het cyber-incident.

Transparantie in gebruikte softwarecomponenten biedt daarnaast aanleiding voor vergelijkend onderzoek. Zo blijkt uit onderzoek van Software Component Analyseleverancier Synopsys dat begin 2023 100% van de software gebruikt in sectoren als luchtvaart en transport één of meerdere Open Source componenten bevatte[19]. Via dezelfde bron is een onderzoek uit 2021 inzichtelijk, waarin wordt vastgesteld dat in ruim 80% van de geanalyseerde software kwetsbare Open Source componenten werden gebruikt. Daarnaast was bij 50% van de software sprake van een licentieconflict, bijvoorbeeld door distributie van componenten waarvoor dit vanuit hun beperkte Open Source licentie niet was toegestaan.

Aanvullende technische controles en validatie bij het gebruik van OSS: CI/CD

OSS leent zich in haar open vorm goed voor het uitvoeren van technische validaties door middel van een CD-straat. Bij professionele softwareontwikkeling is het per definitie van belang een Continuous Integration /

[17] Aan de slag met het kwantificeren van cyberrisico's | Publicatie | Nationaal Cyber Security Centrum (ncsc.nl)

[18] GitHub - NCSC-NL/log4shell: Operational information regarding the log4shell vulnerabilities in the Log4j logging library.

[19] Open source trends from the 2023 OSSRA | Synopsys

Continuous Delivery (CI/CD)-proces[20] uit te voeren. Continuous Integration betreft het frequent aanbrengen van kleine wijzigingen in de code van een centrale productieversie. De aspecten van Continuous Delivery zijn in de context van SSCS het meest relevant. Als onderdeel hiervan wordt namelijk middels een geautomatiseerd proces gecontroleerd of nieuwe code 'release ready' is.

Het valideren en controleren van OSS in een CI/CD omgeving is niet alleen van belang voor het zelf ontwikkelen van OSS maar ook wanneer een organisatie elders ontwikkelde, reeds bestaande OSS, gaat gebruiken. Hierbij kan het open karakter van OSS namelijk worden ingezet om de aanvullende controles uit te voeren, aangezien de broncode daarvoor beschikbaar is.

Er bestaat geen standaard voor de definitie van 'release ready'.*Release readiness* is een security control die door de organisatie zelf in haar controlframework wordt gedefinieerd. De definitie kan worden ingevuld door bijvoorbeeld geautomatiseerde security-tests uit te voeren. CI/CD oplossingen bieden vaak een combinatie van *Static-, Interactive- en Dynamic Application Security Testing* en *Vulnerability Scanning.*

Het is aan te raden om de uitkomsten van dergelijke (geautomatiseerde) onderzoeken terug te delen aan de betreffende OSS-community, indien van toepassing.

Gericht inzetten van code audits

OSS leent zich daarnaast voor het handmatig, menselijk reviewen van softwarecode door het uitvoeren van code audits. Dit kan een passend middel zijn wanneer er volgens uw inschatting sprake is van een verhoogd risico, op een cruciaal belang. Het zelf uitvoeren van een audit geeft uw organisatie veel controle op de aspecten waar naar wordt gekeken, aanvullend op eventuele automatische testen. Verder kan dit middel worden aangewend voor (kleinere) open source projecten die zich in de praktijk (nog) minder hebben bewezen. In de praktijk worden code reviews ingezet voor projecten waar volledig eerstehands inzicht in de code als voorwaardelijke risico-mitigatie wordt gezien, zoals in het voorbeeld van OpenVPN-NL.

Bij het inzetten van dit middel is het uiteraard van belang dat audits worden uitgevoerd door capabel personeel, welke beschikt over de beveiligings-clearance die past bij uw te beschermen belang. Verwerk dergelijke vereisten in uw aanbestedingskaders en selectieprocedures.

Software Security Governance verdient extra aandacht

Wanneer een overheidsorganisatie OSS óf CSS laat (door)ontwikkelen, is het belangrijk dat de organisatie rekening houdt met de eisen uit de Baseline Informatiebeveiliging Overheid (BIO)[21], de verplichte en aanbevolen standaarden van het Forum Standaardisatie[22], security best-practises die passen bij de ontwikkeling van de software en de eventuele vereisten vanuit de OSS-community.

Eigenaarschap van code is bij OSS vaak anders georganiseerd dan CSS, wat effecten kan hebben op juridische aansprakelijkheid. Het is aan te raden om de licentie van een OSS-product en aspecten van aansprakelijkheid te laten onderzoeken door een specialist.

Laat daarnaast het *coordinated vulnerability disclosure* (CVD) proces van het softwareproduct beoordelen. Hieruit moet blijken dat uw organisatie spoedig en

[20] CI/CD - Wikipedia

[21] Baseline Informatiebeveiliging Overheid Cybersecurity - Digitale Overheid

[22] Lijst open standaarden | Forum Standaardisatie

inhoudelijk adequaat wordt geïnformeerd wanneer sprake is van een kwetsbaarheid in de software. Wederkerig moet dit proces ook goed passen wanneer u zelf een nog onbekende kwetsbaarheid meldt bij de ontwikkelaars van de software, bijvoorbeeld naar aanleiding van een (geautomatiseerde) code review. Uit de procesbeoordeling moet daarnaast blijken dat het voor beveiligingsonderzoekers aantrekkelijk is om een kwetsbaarheid bij het betreffende project aan te melden. Zijn er bijvoorbeeld *bug bounties* beschikbaar, die het aantrekkelijk maken om een melding te doen? Betreft het een professioneel, transparant proces, wat aantrekkelijk is om als beveiligingsonderzoeker aan mee te werken?

Open Source en Wet Open Overheid (WOO)

Hoewel deze factsheet zich richt op veilig gebruik van bestaande Open Source software producten, zijn de belangrijkste overwegingen ook van toepassing wanneer u eigen software in Open Source vorm publiceert.

Dit is relevant, omdat van overheidsorganisaties wordt verwacht dat zij zelf ontwikkelde software vrijgeven in de vorm van open broncode. Dergelijke software is vaak ontworpen voor eigen gebruik, niet zozeer voor distributie en hergebruik, maar wordt vrijgegeven in het kader van een transparante overheid. Voor adequate invulling van deze transparantie kan rekenschap worden gegeven aan de belangrijkste uitgangspunten van deze factsheet – deze zijn toepasbaar op zowel softwareafnemer als -producent.

Een recent voorbeeld van een dergelijke publicatie is de vrijgave van de broncode achter de DigiD-App op basis van een EUPL-licentie[23]. Naast de belangrijkste uitgangspunten is bij dit proces extra aandacht besteed aan specifieke zaken die voor publicatie belangrijk waren:

Een grondige review van de code, voorafgaand aan de release. In het kader van transparantie is de code geheel intact gelaten en zijn enkel privacy- en securitykenmerken vervangen door generieke strings zoals SSSSSS (security-kenmerk) en PPPPPP (privacy issue). Een dergelijke aanpak wordt ook voorbereid voor de serverapplicaties voor DigiD[24]. In dit geval past Logius, de onwikkelaar van DigiD, haar beleid stapsgewijs aan om haar code op een veilige en transparante manier beschikbaar te maken onder de WOO.

[23] GitHub - MinBZK/woo-besluit-broncode-digid-app

[24] Kamerbrief over openbaarmaking broncode DigiD-app | Kamerstuk | Rijksoverheid.nl

Handelingsperspectief

1. Hanteer een risico-gebaseerde aanpak als uitgangspunt voor uw security control framework, waarin controls worden opgenomen die u afdoende controle geven over de oorsprong van door u gebruikte softwarecode.
2. Organiseer inzicht in de Software Supply Chain en borg afdoende garanties, zowel in het eigen proces als in de vorm van (juridische, contractuele) garanties, voor veilige uitkomsten.
3. Definieer uw werkwijze omtrent OSS en CSS Supply Chain Security als onderdeel van sourcing-, architectuur-, en security-(governance) kaders.
4. Inventariseer software-afhankelijkheden en gebruikte sub-componenten, monitor deze conform vastgestelde Software Security controls.
5. Valideer OSS softwarecode-kwaliteit via een CI/CD omgeving.
6. Begin en groei: een totaalinzicht en beschikking over alle aanvullende middelen die in dit document zijn besproken vergen een flinke ontwikkeling. Kies een programmatische aanpak om stapsgewijs op het door u gewenste niveau te 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

november ’23