Thema Veiligheid
Tekst Ing. Jan Hendrikx MSc RE CISA CISSP
Beeld 123RF®

Hoe **veilig** is onze software?

Als onze software voedsel was, zouden wij het dan durven opeten? Wat weten wij nu werkelijk over de veiligheid van de software die wij dagelijks binnen onze organisaties gebruiken?

Steeds opnieuw moeten we maar vertrouwen op de goede bedoelingen van een leverancier

We worden steeds afhankelijker van technologie. De opkomst van Internet of Things (IoT), blockchaintechnologie en steeds geavanceerdere apps op onze mobile devices is niet meer te stoppen en de ontwikkelingen gaan razendsnel. De overeenkomst van al deze technologische ontwikkelingen is dat ze gebruikmaken van software. Software vormt de logica die computers en apparaten slim maakt. Het gaat hierbij om computerprogramma's als de apps op onze smartphones tot en met de core applicaties op de servers waar organisaties op steunen.
De steeds verdere integratie van technologie in ons dagelijks leven maakt dat we ook steeds zwaarder steunen op software. Veel van deze software is onzichtbaar voor ons als gebruiker, waardoor we ons vaak onvoldoende bewust zijn van de risico's die het gebruik ervan met zich meebrengt. Dit artikel wil bewustzijn creëren met betrekking tot de beveiligingsrisico's die we als gebruiker lopen bij het gebruik van software.

Totstandkoming van software
Hoe komt veel van de software die we dagelijks gebruiken nu eigenlijk tot stand? Software begint vaak als stukken platte tekst (de zogenaamde broncode, ook wel 'source code' genoemd) op de computer van de ontwikkelaar. Deze stukken tekst bestaan uit een reeks van tekstuele opdrachten, geschreven in een programmeertaal die de uiteindelijke functionaliteit en werking van het computerprogramma bepalen. Als de broncode af is, wordt deze door de programmeur met speciale programmatuur (een 'compiler') omgezet naar een zogenaamde 'executable'. Pas na deze omzetting kan het computerprogramma worden uitgevoerd door een computer.
De eindgebruiker van de software zet een kopie van de executable op zijn computer en kan na het starten van de executable gebruikmaken van de geboden functionaliteit van de software. In de meeste gevallen is het bijzonder lastig om de originele broncode terug te leiden vanuit de executable. Dit komt omdat bij het omzetten van de broncode naar een executable (het compileren) de voor de mens begrijpelijke broncode wordt omgezet naar een reeks machineopdrachten die bedoeld zijn voor de computer.

Closed en open source
Wanneer we software aanschaffen krijgen we meestal alleen de beschikking over de executable. De broncode blijft vanwege het intellectuele eigendom vaak achter bij de organisatie die de software heeft ontwikkeld. Deze software noemen we daarom 'closed source' software. Dit maakt dat we wel gebruik kunnen maken van de software, maar niet kunnen zien wat de kwaliteit van de oorspronkelijke broncode is. Zo is aan een executable in de meeste gevallen niet te zien of het gaat om professioneel ontwikkelde software van hoge kwaliteit of om software die bijvoorbeeld door een beginnende programmeur in India is geschreven. Vaak weten we daardoor weinig tot niets over de kwaliteit en dus de veiligheid van de software die we gebruiken, terwijl het misschien wel de kroonjuwelen van onze organisaties raakt.
In het geval van 'open source' software kan men wel beschikken over de door de programmeur geschreven broncode. Open source software is in de meeste gevallen afkomstig van hobbyisten of organisaties zonder commerciële drijfveer, waarbij de broncode vaak via internet wordt gedeeld. Iedereen kan deze broncode reviewen en dus zien hoe het programma technisch is gebouwd. Natuurlijk dient men dan wel te beschikken over voldoende technische kennis om de kwaliteit en werking van de broncode te kunnen doorgronden. Open source kan helpen bij het beoordelen van de veiligheid van de software. Een eenmaal gereviewde broncode kan men dan zelf omzetten naar een executable, zodat er zekerheid is over de herkomst van de executable.
Wanneer organisaties eigen software ontwikkelen, kan men beschikken over de broncode en kan gestuurd worden op de kwaliteit en veiligheid ervan. Maar zelf software ontwikkelen is kostbaar en complex en daardoor voor veel organisaties geen optie. Dit verklaart de noodzaak om software van derden te gebruiken.

Fouten
Een eigenschap van software is dat het bijna van nature fouten bevat. Dit blijkt wel uit de hoeveelheid updates die we dagelijks op onze apparaten en computers moeten installeren. Deze updates zijn echt niet alleen maar van cosmetische aard, ze lossen vaak ook beveiligingsfouten op. De kans op fouten in open source software is vergelijkbaar met die in closed source software.
Afhankelijk van onder meer de complexiteit van de software, de gebruikte programmeertaal, de inrichting van de ontwikkelomgeving en de kennis en ervaring van de programmeur, is de kans op fouten groter of kleiner. Fouten zijn soms direct zichtbaar, bijvoorbeeld wanneer deze de werking of functionaliteit nadelig beïnvloeden. In andere gevallen blijven fouten onopgemerkt in de programmatuur zitten. Deze fouten kunnen vroeg of laat ontdekt worden door hackers die bewust op zoek zijn naar deze beveiligingslekken om ze vervolgens te misbruiken.

Risico's
De risico's die we met onveilige software lopen kunnen enorm zijn. Datalekken, datacorruptie, 'malware'-infecties, 'ransomware' en cyberaanvallen door hackers zijn maar een aantal voorbeelden van incidenten die kunnen plaatsvinden. Organisaties denken vaak dat ze geen gevaar lopen als ze hun 'patchmanagement' op orde hebben en software-updates tijdig installeren. Patchmanagement is belangrijk, maar wat als er nog geen update bestaat voor een kritieke kwetsbaarheid in een applicatie of besturingssysteem? In veel gevallen zijn organisaties voor hun software-updates volledig afhankelijk van de leverancier. Het besef dat men hierdoor risico's loopt, ontbreekt vaak.
En wat als er bewust een achterdeur in de software is ingebouwd of andere ongewenste functionaliteiten? De sjoemelsoftware in bepaalde dieselauto's laat zien dat dit werkelijk gebeurt. Maar het kan nog veel verder gaan. Technisch gezien kunnen leveranciers van onze besturingssystemen onze servers, laptops en andere mobile devices massaal onklaar maken met één enkele kwaadaardige update. De kans dat dit gebeurt is echter zeer klein, aangezien dit direct tot een verslechterde reputatie zou leiden. Maar het is een reëel scenario in het geval van 'cyber warfare'. Dat is een van de redenen dat de Amerikaanse overheid geen gebruik wenst te maken van Russische antivirussoftware.
Moeten we leveranciers dwingen om inzage te geven in de broncode van de software die we gebruiken? Zelfs al zouden

Wat als er bewust een achterdeur in de software is ingebouwd?

we de broncode van een closed source product op enig moment mogen inzien, wat zou dat dan voor aanvullende zekerheid over de veiligheid bieden? Het blijft een momentopname en bovendien, hoe weten we dat de gereviewde broncode daadwerkelijk deel uitmaakt van de uiteindelijke executable?
Los daarvan kan iedere update opnieuw kwetsbaarheden introduceren en de werking van het product volledig veranderen. Aan een update is vaak niet te zien wat die precies doet. Steeds opnieuw moeten we maar vertrouwen op de kwaliteit en goede bedoelingen van een leverancier. Bijkomend risico is dat updates vaak met verhoogde rechten moeten worden geïnstalleerd, waardoor ze potentieel ook andere zaken op een systeem kunnen beïnvloeden.

Zonder broncode
Natuurlijk bestaat de mogelijkheid om de executables zelf te testen op onveiligheden, zonder de beschikking te hebben over de broncode. Dit doen hackers immers ook. Vaak wordt dan geprobeerd om extreme of ongebruikelijke situaties te creëren om te zien hoe een programma daarop reageert. Indien een programma dan afwijkend gedrag vertoont, bijvoorbeeld door te crashen, kan dat duiden op een kwetsbaarheid die vervolgens verder onderzocht kan worden. Nadeel van dit soort tests is dat deze vaak veel tijdrovender zijn dan broncodereviews. Het maakt de kans om tekortkomingen tijdig te ontdekken aanzienlijk kleiner.

Security Development Lifecycle
Hoe kunnen we dan wel zekerheid krijgen over de veiligheid van onze software? Men kan de totstandkoming van software in sommige opzichten vergelijken met de bereiding van voedsel. Wanneer voedsel onzorgvuldig wordt bereid, loopt degene die het voedsel opeet een groter risico om ziek te worden. Door de bereiding ervan onder gecontroleerde omstandigheden plaats te laten vinden, kunnen problemen worden voorkomen. Zo is het ook met software. Om de veiligheid van software te beoordelen zal in de 'keuken' gekeken moeten worden.
Een manier om die keuken op orde te krijgen en te houden, is via het inrichten van een 'security development lifecycle' (SDL) proces. Zo'n proces zorgt dat in alle fasen van de ontwikkeling het aspect beveiliging wordt meegenomen en beoordeeld. Inmiddels zijn er al een aantal onafhankelijke partijen die het SDL-proces kunnen reviewen en certificeren, zodat ook leveranciers van software kunnen aantonen dat zij op een verantwoorde manier veilige software maken.

Rol van de auditor
Internal auditors moeten zich vaker de vraag stellen of de organisatie wel op de door haar gebruikte software kan steunen. Dat is lastig, want hoe diep moet je daarbij gaan? Dat hangt helemaal af van waarvoor en op welke wijze de organisatie de software gebruikt en welke informatie deze verwerkt. Vervolgens kan de auditor de vraag stellen of voldoende zekerheid bestaat over de veiligheid van de gebruikte software. Misschien zijn aanvullende zekerheden nodig, zoals certificering of een broncode review. Wordt binnen de eigen organisatie software ontwikkeld, dan kan de auditor zelf het interne ontwikkelproces onderwerpen aan een audit. Als dit niet het geval is, dan zal de auditor zekerheid over de totstandkoming van software moeten vragen aan de leverancier. Gezien het alsmaar toenemende belang van software in combinatie met de steeds strenger wordende regelgeving rondom de privacy, een vraag die we bijna verplicht zijn om te stellen! <<

Jan Hendrikx is ondernemer op het gebied van technische informatiebeveiliging. Hij voert technische beveiligingsonderzoeken uit en adviseert organisaties over de wijze waarop zij hun beveiliging tegen onder meer cybercriminaliteit kunnen optimaliseren.
jan@hendrikx-itc.nl