

Drie weken om met AppSec van start te gaan - Het meest eenvoudige plan om uw beveiligingsprofiel te verbeteren

05 januari 2021, 01:00

Nick Boucart
Tatiana Galibus

'Shift security left' is een populair paradigma in de industriesector, dat makkelijk te begrijpen is maar bij de implementatie niet zo vanzelfsprekend lijkt. Voor de toepassing van deze stelling is meer dan alleen het gebruik van technologie nodig: het impliceert een verschuiving in de cultuur, een geïntegreerde benadering van de applicatiebeveiliging en een permanent leerproces. De meeste start-ups willen de theorie ervan wel overnemen, maar botsen op obstakels bij de praktische toepassing.

Misschien zijn de bestaande strategieën wel op maat gemaakt voor bedrijven met een bepaald beveiligingsprofiel, maar er bestaan geen gemeenschappelijke richtlijnen over de manier om de applicatiebeveiliging aan te pakken, vooral wanneer de start-ups over beperkte middelen en expertise beschikken. Welke eenvoudige maatregelen moeten worden getroffen om een duidelijk optimalisatieplan voor de maturiteit van de beveiliging uit te werken en klaar te zijn om de vragen van de klanten over vertrouwen en beveiliging te beantwoorden?

Gedreven door de passie om een pragmatische oplossing voor start-ups te vinden die echt werkt, hebben we bij Sirris bijna drie maanden aan onderzoek en brainstorming gedaan. Het resultaat? *Het eenvoudigste 3-wekenplan om uw beveiligingsprofiel te verbeteren*. Het plan bouwt voort op de



In een vorig artikel lichtten we de stappen toe die men gedurende de eerste week moet volgen. Nu komen de stappen aan bod die men in de tweede week van het meest eenvoudige plan voor een beter beveiligingsprofiel moet volgen.

Tweede week: de kwaliteitsdrempels bepalen en statische codeanalysetools in uw bouwketen integreren

De tweede week praktijk spitst zich toe op uw eerste stappen naar *beveiligingsautomatisering*. Er bestaan honderden commerciële en opensourcetools voor de veiligheidsanalyse van een applicatie/website. Zo zijn er onder meer tools die de code en de afhankelijkheden ervan analyseren (en niet de actieve applicatie); deze worden **SAST-tools voor statische applicatiebeveiligingstests (Static Application Security Testing)** genoemd. Dergelijke tools worden in verschillende categorieën ingedeeld:

1. Codescanners

De scanners toetsen uw code af tegen de ingebouwde coderingsregels en melden eventuele problemen met de codekwaliteit of zwakke plekken in de beveiliging. Deze tools kunnen honderden regels voor elke programmeertaal en verschillende scanalgoritmes hebben. Sommige zijn op de beveiliging gericht en bieden dus een grondige analyse van de zwakke plekken in de beveiliging. Andere zijn op de codekwaliteit gericht en melden dus mogelijke problemen met de codering of onnauwkeurigheden. We raden u aan om met een van de volgende tools van start te gaan:

SonarScanner	Voert een scan uit naar problemen in zowel de beveiliging als de codekwaliteit. Er bestaat een commerciële en een opensourceversie van en de kan als plug-in in uw ontwikkelingsomgeving wor geïntegreerd, waardoor hij gebruiksvriendelijk is.
Fortify	Een opensourcetool die op de beveiliging is geric tool is eenvoudig in de ontwikkelingsomgeving er CI/CD-pijplijn geïntegreerd.

2. Softwaresamenstellinganalysetools (SCA - Software Composition Analysis)

SCA-tools kunnen codeafhankelijkheden en componenten van derden scannen op bugs en zwakke plekken in de beveiliging. Zelfs al voldoet uw code aan de veiligheidseisen, kan bijvoorbeeld een vraag naar onbeveiligde functies, zoals Java Random.NextInt() een bron van kwetsbaarheid vormen. Vaak denken we er niet aan beveiligde API's en bibliotheken te gebruiken. We raden de volgende tools voor het scannen op afhankelijkheden aan:

OWASP DependencyCheck	Voert een scan uit naar onveilige afhankelijkheden; geïntegreerd in SonarQube.
Fortify	Voert een scan uit naar onveilige afhankelijkheden en bibliotheken; een Fortify codescan plug-in.
Retire.js	Controleert het gebruik van onveilige bibliotheken en componenten in Java.
Bundler-audit	Controleert op kwetsbare versies van Ruby-componenten.
Gemnasium/GitLab security scan	Monitort de componentversies in verschillende programmeertaken en meldt wijzigingen.Momenteel is deze tool geïntegreerd in de GitLab CI/CD automatisering-pijplijn.

3. Tools voor het detecteren van geheimen

Vaak sluiten ontwikkelaars ongewild geheimen, API-tokens en referenties in repositories op afstand in. In dat geval worden gevoelige gegevens makkelijk blootgelegd en kunnen ze door tegenstanders worden gebruikt om de identiteit te stelen en ongeoorloofde toegang te krijgen tot specifieke resources. Tools voor het detecteren van geheimen kunnen de code scannen en de aanwezigheid van sleutels, tokens en referenties scannen. We raden de volgende tools aan (de meeste zijn in de overeenkomstige CI/CD-omgeving geïntegreerd):

GitGuardian	Automatisch geïntegreerd in de GitLab repositoryomgeving.
GittyLeaks	Automatisch geïntegreerd in de GitHub repositoryomgeving.
Git-secrets	Voorkomt dat de ontwikkelaars geheimen in de G repository insluiten.
Truffle Hog	Deze tool scant de repositories en alle branches op gevoelige informatie.

4. Rooktesten en tools voor containeranalyse

Rooktesten of Build Verification Testing (BVT) brengen simpele defecten aan het licht die voldoende ernstig zijn om een softwarerelease te weigeren. Tools voor het scannen van Docker-containers zijn essentieel om uit te maken welke bibliotheken in uw container kwetsbaar zijn. Deze tools detecteren de onveiligheden en bugs in Docker-afbeeldingen vóór de code wordt toegepast. We raden de volgende tools voor rooktesten en containerscans aan:

Clair-scanner	Een tool voor het scannen van containers, die in GitLab CI/CD-pijplijn is geïntegreerd.
Trivy	Scant containerkwetsbaarheden door problemen OS-pakketten en applicatieafhankelijkheden te detecteren.
Twistlock	Biedt containerveiligheid over een volledige cyclu geïntegreerd in het cloud-based beheersysteem kwetsbaarheden.

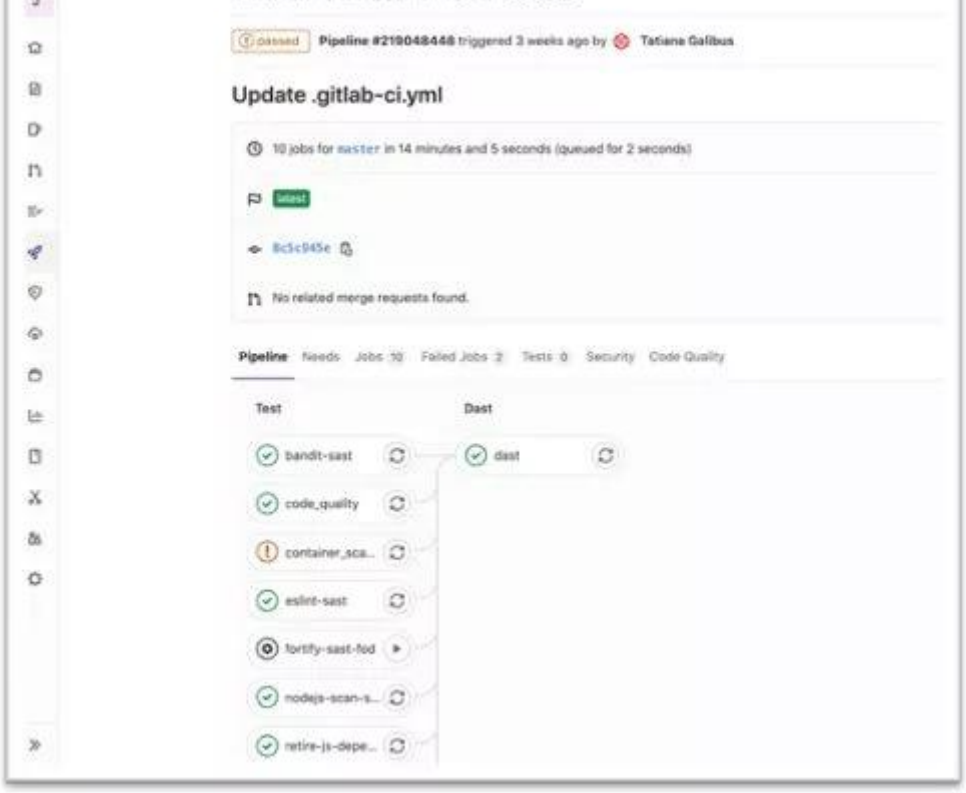
5. Kwetsbaarheidsbeheerders

Kwetsbaarheidsbeheer is een essentieel onderdeel van de continue integratie; het vereenvoudigt immers het continu monitoren, identificeren en voorkomen van risico's voor de code, afhankelijkheden, containers, afbeeldingen en hosts in hun omgeving. Vaak worden SAST-tools naast kwetsbaarheidsbeheerders ingezet om diepere inzichten in de veiligheidsproblemen te verkrijgen. We raden de volgende kwetsbaarheidsbeheerders aan:

SonarQube server	Geeft alle gedetecteerde problemen weer, per gr van ernst en type in een project dashboard. Maal mogelijk de kwetsbaarheden op te sporen, af te b en te analyseren.
DefectDojo	Een platform voor kwetsbaarheidsbeheer om de van meerdere veiligheidsscanners te integreren e organiseren.
Tenable.io	Een tool voor kwetsbaarheidsbeheer die in GitLa geïntegreerd. Biedt een automatisch dashboard e rapporten over bugs in de beveiliging.

Een van de belangrijkste pluspunten van SAST-tools is de brede dekking van programmeertalen en ontwikkelingsplatforms. Ze zijn relatief eenvoudig toe te passen en te integreren. Ondanks een groot aantal vals-positieven, maken SAST-tools het mogelijk zowat de helft van de bugs te identificeren en zijn ze in het bijzonder efficiënt voor het verbeteren van de codekwaliteit en het detecteren van terugkomende veiligheidsproblemen.

SAST-tools vormen de grondslag van een applicatiebeveiligingspijplijn. De AppSec-pijplijn is een keten van essentiële veiligheidstools die in alle stadia van de levenscyclus van de DevSecOps-softwareontwikkeling is geïntegreerd. SAST-tools bieden een continue veiligheid in de stadia van ontwerpen, coderen, bouwen en valideren.



Voorbeeld van een GitLab-pijplijn, SAST-tools en testfase

Benieuwd naar meer informatie over dit onderwerp en andere stappen? Hebt u andere praktische vragen over applicatiebeveiliging? Neem contact met ons op via security@sirris.be.



]]>

Auteurs



Nick Boucart



Tatiana Galibus