

▮ Øvelse 28 – Detektering af sårbare afhængigheder med Dotnet CLI-værktøjet

▮ Information

Formålet med denne øvelse er at introducere tredjeparts sårbarheder og CVE'er.

Det meste software i dag bygger på kode, som andre har udviklet tidligere. Disse eksterne komponenter importeres som pakker eller biblioteker.

Når man bruger 3. parts biblioteker, importerer man kode, som man ikke selv har udviklet, og man har ikke altid indflydelse eller indsigt i. Dette kan være problematisk, hvis der opstår en sårbarhed. CVE-databaser som [CVE.org](https://www.cve.org) indeholder information om kendte sårbarheder i software.

En opgørelse over alle tredjepartsbiblioteker i et softwareprojekt kaldes ofte en *Software Bill of Materials (SBOM)*.

I .NET importeres eksterne biblioteker som NuGet-pakker. Du kan se listen over pakker i projektfilen, som altid ender på `.csproj`.

I denne øvelse skal du anvende CLI-værktøjet `dotnet` til at scanne et .NET-projekts NuGet-pakker for sårbarheder.

Du skal klonere repoet <https://github.com/tobyash86/WebGoat.NET>, som er et bevidst sårbart ASP.NET Core-projekt.

▮ Instruktioner

1. ▮ Gå ind i mappen til det klonede repository.
2. ▮ Følg instrukserne i denne [tutorial](#) for at udføre en sårbarhedsskanning med Dotnet CLI (se afsnittet om Dotnet CLI).
3. ▮ Følg linket til den fundne sårbarhed og noter CVE-nummeret øverst på siden.
4. ▮ Slå CVE'en op på [CVE.org](https://www.cve.org) og undersøg, om der findes mere information om sårbarheden.

▮ *Note:* I professionelle udviklingsmiljøer køres sådanne sårbarhedsskanninger ofte automatisk som en del af build- og release-pipelines. Dette sikrer, at nye sårbarheder opdages tidligt, og at sårbar kode ikke deployes i produktion.

1. Brug `--include-transitive` flaget for at se underliggende (transitive) sårbarheder (se tutorial for detaljer).

Refleksionsspørgsmål

1. Hvorfor er det vigtigt at være opmærksom på sårbarheder i tredjepartsbiblioteker?
2. Hvordan kan brug af et værktøj som Dotnet CLI hjælpe med at styrke sikkerheden i softwareprojekter?
3. Hvilke udfordringer kan der være ved at håndtere transitive afhængigheder?
4. Hvordan hænger SBOM sammen med risikostyring i softwareudvikling?
5. Bør en skanning for 3. Parts sårbarheder ekskveres løbende?
6. Bør en skanning for 3. Parts sårbarheder være en manuel, eller automatisk process?

Kobling til relevante standarder og principper

Element i øvelsen	Relateret standard / princip	Kommentar
Sårbarhedsscanning	ISO/IEC 27002:2022, afsnit 8.8. (Styring af tekniske sårbarheder, særligt stykke d. og f.)	Anbefaler løbende scanning og håndtering af kendte sårbarheder i softwarekomponenter.
	OWASP Application Security Verification Standard (ASVS) Kontrol 15	Krav om håndtering af tredjepartskomponenter og kendte sårbarheder i applikationer.
	Secure by Design (Johnsson et al.)	Fremhæver vigtigheden af at kende og minimere risiko ved tredjepartssoftware.

Proaktiv scanning og opdatering af tredjepartsbiblioteker er afgørende for at opretholde et sikkert softwaremiljø.