Module 7: Assignment: Secure Software Supply Chain with SBOMs

First we clone NG911: git clone https://github.com/tamu-edu/ng911-dev.git

```
@JorgeVanco ➜/workspaces/eng298-fa25-mod7-sbom-lab1 (main) $ git clone https://github.com/tamu-edu/ng911-dev.git
Cloning into 'ng911-dev'...
remote: Enumerating objects: 2357, done.
remote: Counting objects: 100% (1557/1557), done.
remote: Compressing objects: 100% (854/854), done.
remote: Total 2357 (delta 861), reused 1321 (delta 688), pack-reused 800 (from 4)
Receiving objects: 100% (2357/2357), 8.62 MiB | 23.93 MiB/s, done.
Resolving deltas: 100% (1166/1166), done.
```

Then we generate an SBOM in SPDX format (Syft):

```
@JorgeVanco ➜/workspaces/eng298-fa25-mod7-sbom-lab1 (main) $ cd ng911-dev
@JorgeVanco ➜/workspaces/eng298-fa25-mod7-sbom-lab1/ng911-dev (main) $ syft . -o spdx-json > ../deliverables/sbo
m_syft_spdx.json
 ✓ Indexed file system                                                                                    .
 ✓ Cataloged contents                        cdb4ee2aea69cc6a83331bbe96dc2caa9a299d21329efb0336fc02a82e1839a8
   ├── ✓ Packages                        [107 packages]
   ├── ✓ File digests                    [3 files]
   ├── ✓ File metadata                   [3 locations]
   └── ✓ Executables                     [0 executables]
[0000]  WARN no explicit name and version provided for directory source, deriving artifact ID from the given path
@JorgeVanco ➜/workspaces/eng298-fa25-mod7-sbom-lab1/ng911-dev (main) $
```

Generate a CycloneDX SBOM (Trivy):

```
@JorgeVanco ➜/workspaces/eng298-fa25-mod7-sbom-lab1/ng911-dev (main) $ trivy fs . --format cyclonedx --output ..
/deliverables/sbom_trivy_cdx.json
2025-12-15T16:14:32Z    INFO    "--format cyclonedx" disables security scanning. Specify "--scanners vuln" explic
itly if you want to include vulnerabilities in the "cyclonedx" report.
2025-12-15T16:14:33Z    INFO    [python] Licenses acquired from one or more METADATA files may be subject to addi
tional terms. Use `--debug` flag to see all affected packages.
2025-12-15T16:14:33Z    INFO    [npm] To collect the license information of packages, "npm install" needs to be p
erformed beforehand     dir="test_suite/test_files/_old/TPLan_Config/VS_Code/node_modules"
2025-12-15T16:14:33Z    INFO    Number of language-specific files       num=2
```

Trivy reported 106 components and Syft reported 108 packages. However, Trivy has two more components in the metadata, so that makes a total of 108 components as well.

```
    syft = json.load(open("sbom_syft_spdx.json"))
    print("Number of packages by syft:", len(syft['packages']))
[16]  ✓ 0.0s
...  Number of packages by syft: 108


    trivy = json.load(open("sbom_trivy_cdx.json"))
    print("Number of compontents by trivy:", len(trivy["components"]))
[17]  ✓ 0.0s
...  Number of compontents by trivy: 106
```

The formats have different fields and ways of counting the components:

```
    syft_keys = list(syft.keys())
    print("Syft fields:")
    syft_keys
```

[21] ✓ 0.0s

```
Syft fields:

['spdxVersion',
 'dataLicense',
 'SPDXID',
 'name',
 'documentNamespace',
 'creationInfo',
 'packages',
 'files',
 'relationships']
```

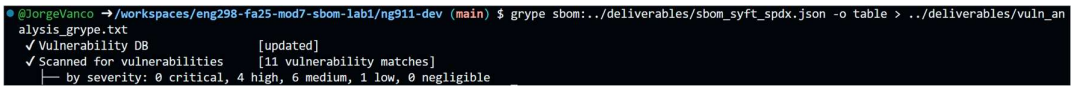
```
    trivy_keys = list(trivy.keys())
    print("Trivy fields:")
    trivy_keys
```

[20] ✓ 0.0s

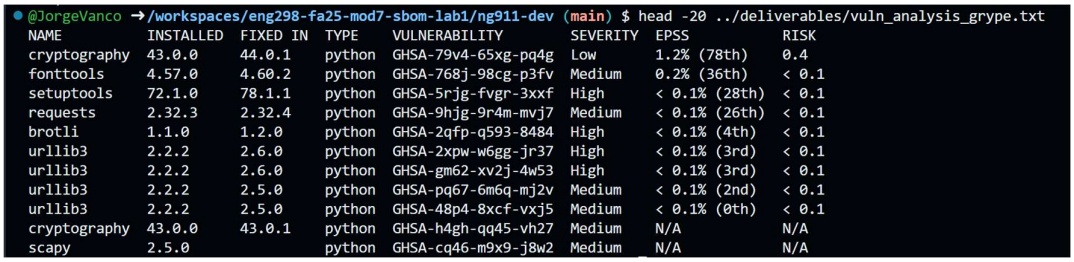
```
Trivy fields:

['$schema',
 'bomFormat',
 'specVersion',
 'serialNumber',
 'version',
 'metadata',
 'components',
 'dependencies',
 'vulnerabilities']
```

Then we feed the Syft SBOM into Grype to discover CVEs:

grype sbom:../deliverables/sbom_syft_spdx.json -o table > ../deliverables/vuln_analysis_grype.txt



The following image is the output we got:



```
● @JorgeVanco →/workspaces/eng298-fa25-mod7-sbom-lab1/ng911-dev (main) $ head -20 ../deliverables/vuln_analysis_grype.txt
  NAME          INSTALLED  FIXED IN  TYPE    VULNERABILITY        SEVERITY  EPSS           RISK
  cryptography  43.0.0     44.0.1    python  GHSA-79v4-65xg-pq4g  Low       1.2% (78th)    0.4
  fonttools     4.57.0     4.60.2    python  GHSA-768j-98cg-p3fv  Medium    0.2% (36th)    < 0.1
  setuptools    72.1.0     78.1.1    python  GHSA-5rjg-fvgr-3xxf  High      < 0.1% (28th)  < 0.1
  requests      2.32.3     2.32.4    python  GHSA-9hjg-9r4m-mvj7  Medium    < 0.1% (26th)  < 0.1
  brotli        1.1.0      1.2.0     python  GHSA-2qfp-q593-8484  High      < 0.1% (4th)   < 0.1
  urllib3       2.2.2      2.6.0     python  GHSA-2xpw-w6gg-jr37  High      < 0.1% (3rd)   < 0.1
  urllib3       2.2.2      2.6.0     python  GHSA-gm62-xv2j-4w53  High      < 0.1% (3rd)   < 0.1
  urllib3       2.2.2      2.5.0     python  GHSA-pq67-6m6q-mj2v  Medium    < 0.1% (2nd)   < 0.1
  urllib3       2.2.2      2.5.0     python  GHSA-48p4-8xcf-vxj5  Medium    < 0.1% (0th)   < 0.1
  cryptography  43.0.0     43.0.1    python  GHSA-h4gh-qq45-vh27  Medium    N/A            N/A
  scapy         2.5.0                python  GHSA-cq46-m9x9-j8w2  Medium    N/A            N/A
```

Here is a table with the top 5 vulnerabilities with highes severity found. I had to use GitHub Advisory Database to get the CVE from the GHSA:

CVE	Severity	Component	Version	Comment
CVE-2025-47273	High	setuptools	72.1.0	Path traversal in PackageIndex allows arbitrary file writes and potential RCE.
CVE-2025-6176	High	brotli	1.1.0	Denial of Service (DoS) via "decompression bomb" causing memory exhaustion.
CVE-2025-66471	High	urllib3	2.2.2	Streaming API improperly handles compressed data, leading to excessive resource consumption.
CVE-2025-66418	High	urllib3	2.2.2	Denial of Service (DoS) possible via unbounded recursion in the decompression chain.
CVE-2025-66034	Medium	fonttools	4.57.0	XML injection and path traversal in varLib can lead to arbitrary file writes and RCE.

I searched **CVE-2025-47273** in the National Vulnerability Database:

A path traversal vulnerability in the setuptools PackageIndex allows attackers to overwrite arbitrary files on the filesystem, which can potentially escalate to remote code execution.