A6: Security Misconfiguration +

A9: Components with Known Vulnerabilities

OPGAVE 1

**A6, Using default settings**

1. **See whether you can discover the following properties of the application (not all are necessary security-problems). Use the GUI provided by the application (as a start), Postman, nmap and obviously your browser's Developer Tools, when probing the app:**

**Link** 🡪[link points](http://sec-dat-demo.surge.sh/)

**OS** = Linux / nginx

**Server Architecture (Come up with a “guess” and provide arguments for your suggestion)**

Nginx/tomcat

**Server(s) =** Linux, nginx, tomcat 🡪 response headers

Find serverens OS (Operativ system) 🡪 Network 🡪 Headers

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

**Programming Language** = JAVA 🡪 bad request (stack trace)

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

**Can you discover the client technologies used?** = REACT 🡪 node\_modules

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

**If you can make a successful login, can you: discover the algorithm used to “protect” the token, the lifetime of the token, the role, assigned to you by the system?**

* JSON web token (JWT-token)
* Algoritme: HS256 (hash algoritme)
* Lifetime: 30 min
* Role: user/admin
* Translate token 🡪 <https://jwt.io/>

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

**How/where is the token stored by the client**

* Applikation 🡪 Local Storage

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

**Can you determine/guess(must be qualified) whether front-end, REST back-end and Database is running on the same or on different servers?**

* De kører på forskellige servere
* IP’er findes i Network 🡪 headers 🡪 General 🡪 Remote Address
* **Frontend på surge.sh** 🡪 IP: 138.197.235.123

Et billede, der indeholder skærmbillede, fugl, blomst

Automatisk genereret beskrivelse

* **Backend på: dat-security.dk/secDemoA6/api/** 🡪 IP: 46.101.227.238

Et billede, der indeholder skærmbillede, fugl

Automatisk genereret beskrivelse

**Can you determine which database is used by the backend?**

For at scanne: Åben Kali Linux *🡪 Terminal 🡪 skriv: nmap -v -A ”IP\_ON\_BACKEND”*

* MySQL 🡪 port 3306

Et billede, der indeholder tekst, sort, sidder, bord

Automatisk genereret beskrivelse

**Have you discovered any unnecessary features which are enabled or installed (e.g. unnecessary ports, services, pages, accounts, or privileges)**

* Port 3306 bør ikke være åben, fordi requests til databasen bliver lavet fra samme server og ikke udefra.
* **Who owns the domain used for the server?**

<http://whois.domaintools.com/>



<http://whois.domaintools.com/dat-security.dk>

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

* **Is the server hosted “privately”, by a cloud provider, or …..?**

Serveren er sat op fra Digital Ocean, gennem en privat bruger.

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

* **… Can you detect/discover more properties of the application than those suggested above?**

OPGAVE 2

**A6, Security Headers**

1. **Enter this link** [**http://security-headers.dat-security.dk/**](http://security-headers.dat-security.dk/) **in a browser, and explain shortly the purpose of all response headers, related to security.**

Network 🡪 Headers 🡪 repsonse header

Nedenstående er en række security headers og bruges til at afgøre hvordan en browser skal opføre sig, når den skal håndtere en sides indhold.

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

**X-Frame-Options:** SAMEORIGIN

* Indikerer hvorvidt en browser skal ”render” en hjemmeside.
* X-frame angivet med ”sameorigin” tillader kun at vise den egentlige side mulighed for at ”frame”/vise indholdet.
* X-frame-options bruges til at indikere om en side må gengives i et <frame> eller <iframe>. Websites kan bruge denne security header, for at undgå clickjacking, ved at sikre at deres indhold ikke kan indlejres i andre sider.

**HTTP-Strict-Transport-Security:** max-age=15552000; includeSubDomains

* IncludeSubDomains er optional og betyder at dette også gælder for alle sidens sub domæner.
* Lader et website fortælle browsere, at det kun må tilgås ved hjælp af HTTPS i stedet for at bruge http.
* Expire-time i sekunder som browseren skal huske at denne side kun må tilgås ved brug af HTTPS
* **Attack scenarie:** En man-in-the-middle forsøger at opfange offerets netværkstrafik, ved at anvende et ugyldigt certifikat, i håb om at offeret accepterer dette. Headeren HSTS tillader ikke offeret at anvende et ugyldigt certifikat.

**X-Content-Type-Options:** nosniff

* Browseren gætter på content type (MIME type), hvis det ikke er angivet.
* Ved at definere denne header, beder vi browseren om ikke at prøve at gætte (MIME type sniffing), men blot vise det som ”plain text”, medmindre content type er angivet (kunne f.eks. være HTML)

**Content-Security-Policy:** default-src 'self'; style-src 'self' maxcdn.bootstrapcdn.com; font-src 'self' https: data:

* Hvad må klienten kunne tilgå?
* Self indikere at alt indhold skal komme fra sidens egen ORIGIN (også sub domains)
* Beskytter mod XSS og data injection angreb.
* Brugerdefineret policy, bør inkludere ” default-src” eller ”script-src” og ”style-src”.

**X-XSS-Protection:**

* En security header, der kan anvendes på browsere, som Safari, Chrome og Internet Explore, der stopper en side i at loade, hvis der detekteres ”reflected-cross-site-scripting”.
* Kan sættes til enten 0 eller 1.

1: mode = block (stopper browseren I at ”rende” siden).

1: report = <reporting-uri>

0: disabled.

1: enabled og anvender sanitizing.

1. **Setup a simple web server (Tomcat, Nginx, Express ….) which should set most of the security headers for all requests.**

*This is probably simpler than what you might think. With node + Express, you can create such a server in less than six lines of code ;-)*

*If you are using Tomcat (or Express), behind a Nginx-reverse proxy, you can do this either in your code (Tomcat, Express) or, let Nginx add the headers.*

**Gå ind I terminalen og skriv:** *sudo nano /etc/nginx/nginx.conf*

Under http level, tilføj **security headers.**

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

**Her kan man se de er tilføjet:**

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

## OPGAVE 3

## **A9 Components with Known Vulnerabilities**

How can we (you) ensure that our maven dependencies do not contain Known Vulnerabilities?

1. **Google this topic an see what kind of tools you can find, the following are suggestions:**

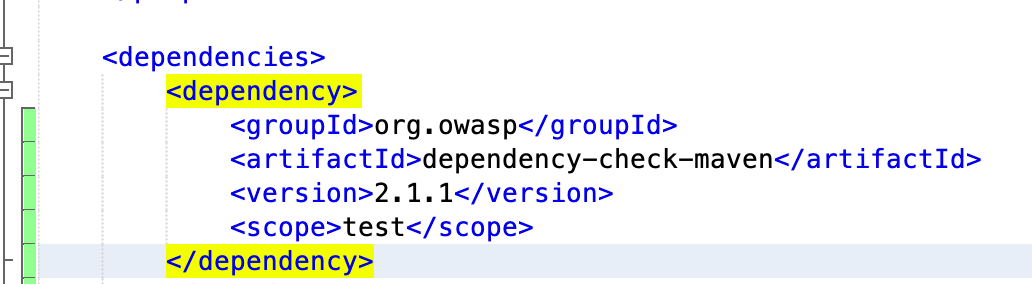
* <https://snyk.io/vuln/?packageManager=all>
* <https://www.owasp.org/index.php/OWASP_Dependency_Check>

1. **Use one or more of the tools/strategies found above and use them to check some of your previous Java/Maven projects (for example the backend seed from your 3. Semester CA3)**

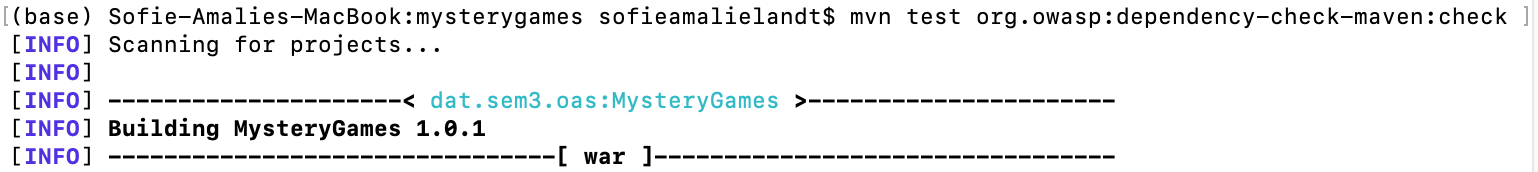
En strategi til at finde ”known vulnerabilites”:

**Dependency health**

Tilføj denne dependency i pom.xml



For at køre OWASP dependency check, skriv i terminalen:



Resultatet:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Ved at anvende snyk.io, kan man på samme måde som ovenover, teste for known vulnerabilites”, ved at forbinde med GitHub:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

1. **If you are following Python or JavaScript come up with a similar strategy for Python/JavaScript dependencies**