Komposition – A6 og A9

**Agenda:**

1. Forklar om A6 Security Misconfiguration Risk
2. Vis eksempel på hvor mange oplysninger man kan finde på en fejlkonfigureret applikation.
3. Forklar og demonstrer Security Headers.
4. Forklar om A9 og demonstrer et dependency check.

**Explain briefly about the A6 Security Misconfiguration Risk, including topics from:**

Hackere vil ofte udnytte eller forsøge at få adgang til default brugere, ubrugte sider, ubeskyttede filer eller mapper, for at få uautoriseret adgang eller viden om systemet/programmet. Et værktøj som nmap og nikto, er automatiserede scannere, som er effektive til at detektere fejlkonfigurationer. Disse fejlkonfigurationer, kan ske i alle lag af applikationen, som f.eks. webserveren eller databasen.

**Is the Application Vulnerable**

En applikation kan være sårbar, hvis applikationen:

* Har manglende sikkerheds hardening. Det er en proces til at mindske sårbarheder på applikationens overflade, f.eks. ændre default adgangskoder, lukke unødvendige porte udefra eller fjerne eller disable unødvendige services.
* Har unødvendige funktioner aktiveret eller installeret, unødvendige porte, services, sider, brugerkonti.
* Har aktive brugere med default adgangskoder, som er uændret.
* Har error handling, der afslører stack trace eller andre overdrevne informative error beskeder til brugere.
* Har en server, der ikke sender security headers eller at disse ikke er sat til sikre værdier.
* Har opgraderede systemer, hvor de seneste sikkerhedsfunktioner ikke er aktiveret, eller ikke er konfigureret sikkert.
* Har software som er forældet eller sårbart.

Uden kontinuerlig sikkerheds konfiguration, er systemer mere i højere risiko for at blive hacket.

**How to Prevent**

Sikre installations processer, der bør implementeres, inkludere:

* Kontinuerlig sikkerheds hardening, som gør det hurtigt og nemt at deploy nye miljøer, som er korrekt lukket.
* At alle miljøer automatisk er konfigureret ens, med forskellige credentials, brugt i de forskellige miljøer.
* En minimal platform, uden unødvendige funktioner, komponenter etc.
* En segmenteret applikations arkitektur, der giver en effektiv og sikker adskillelse af komponenterne, f.eks. at have sin database på en anden server, end selve applikationen.
* At sende security headers til klienter.
* En automatiseret proces, til at bekræfte effektiviteten af konfigurationerne og indstillingerne i alle miljøer.

**Example Attack Scenarios**

1. Applikationsserveren indeholder eksempelapplikationer, der ikke fjernes fra produktionsserveren. Disse eksempelapplikationer har kendte sikkerhedsfejl, som hackeren udnytter til at kompromittere serveren. Hvis en af disse applikationer har administrator rettigheder, og default konti ikke er ændret, kan hackeren logge ind med default adgangskode og overtage serveren.
2. Hvis konfigurationen af applikationsserveren tillader detaljerede fejlmeddelelser, f.eks. stack trace, som returneres til brugerne, kan dette afsløre potentielt følsomme oplysninger eller underliggende fejl, såsom komponentversioner, der vides at være sårbare.

**Explain and demonstrate a number of the new Security Headers used by modern browsers, with emphasis on how they can increase security and the problem they protect against**

<https://security-headers.dat-security.dk/>

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 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.

**Demonstrate shortly how many properties of a sample (misconfigured) application a hacker can discover.**

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

**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.

**Explain, as many as you can come up with, Security Misconfigurations, made deliberately throughout this semester, to make it possible to demonstrate “real attacks”**

* Default brugere med default oplysninger
* Stack trace I forbindelse med Exceptions
* Unødvendige features som gør det muligt at tilgå serveren (f.eks. åben database port)
* Serverere der ikke inkluderer security headers.

**Explain briefly about the OWASP A9 threat**

Selvom det er let at finde allerede skrevne udnyttelser til mange kendte sårbarheder, kræver andre sårbarheder koncentreret indsats for at udvikle en brugerdefineret udnyttelse.

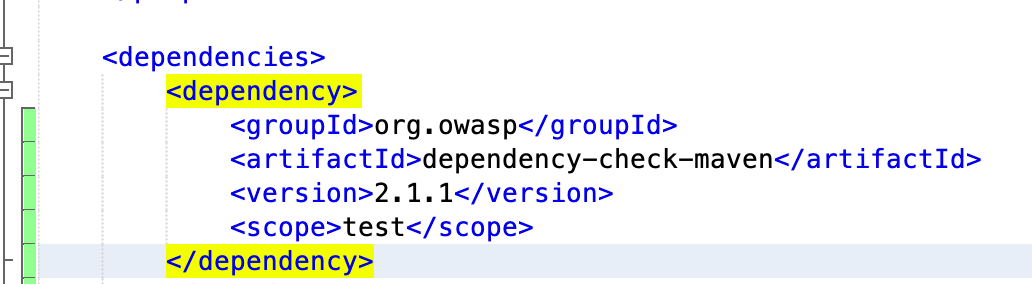
Kendte sårbarheder er sårbarheder, der kan blive opdaget i open source-komponenter og er offentliggjort. En sårbarhed udnyttes af hackere, der kan finde dokumentationen for open-source-komponenten der anvendes. Ifølge OWASP er problemet med at bruge komponenter med kendte sårbarheder meget udbredt.

**Provide and demonstrate practical examples for how to ensure that our maven dependencies do not contain Known Vulnerabilities? (Feel free to replace Java/maven above with examples from other technologies such as JavaScript, Python …)**

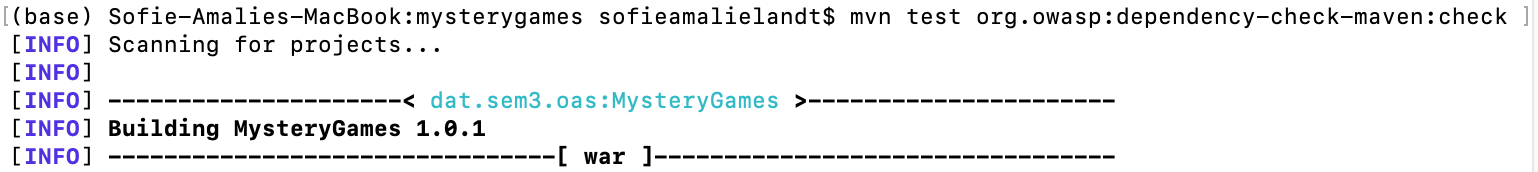
En strategi til at finde ”known vulnerabilites”:

**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