Mange tig der skal konfigurere når vi opsætter en applikation. Eks. Da vi skulle sætte vores droplet op med Tomcat og SSL. Man kan let komme til at efterlade bagdøre åbne for hackere at misbruge.

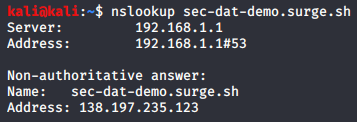
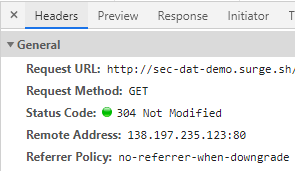
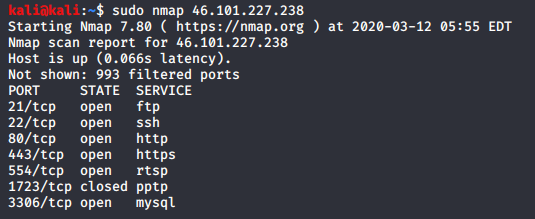
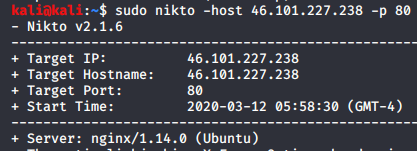
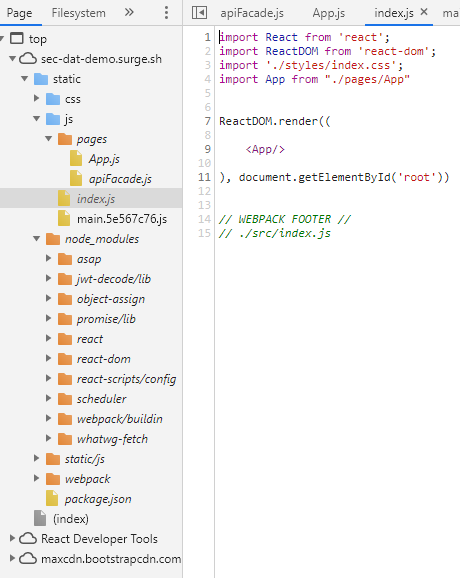
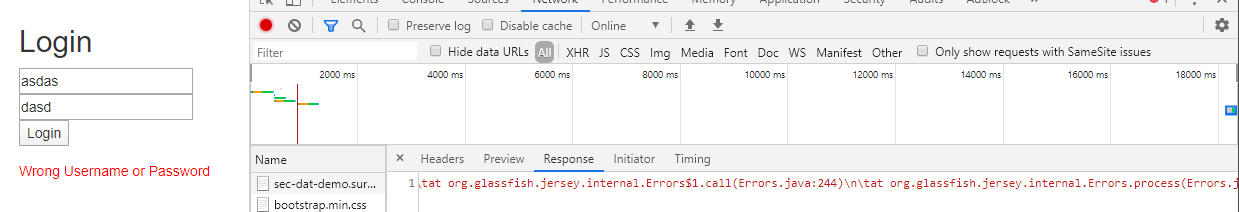
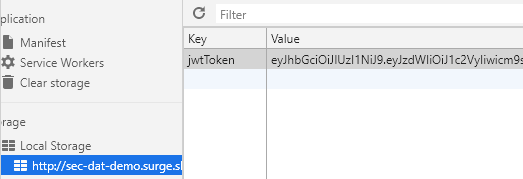
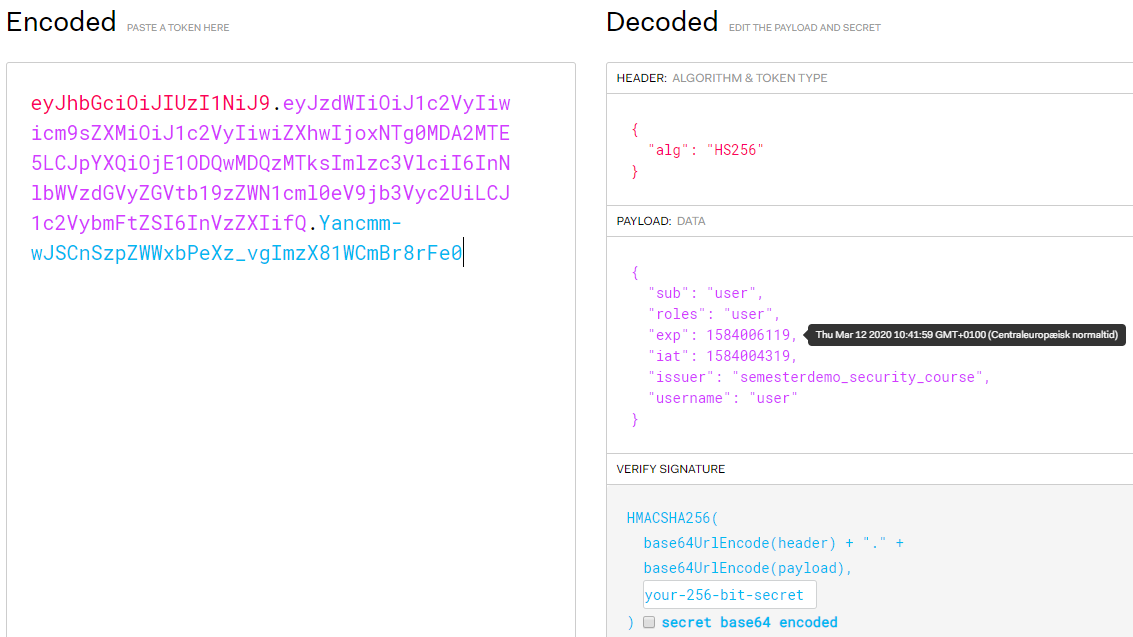
The application might be vulnerable if the application is:

* Missing appropriate security hardening across any part of the application stack, or improperly configured permissions on cloud services.
* Unnecessary features are enabled or installed (e.g. unnecessary ports, services, pages, accounts, or privileges).
* Default accounts and their passwords still enabled and unchanged.
* Error handling reveals stack traces or other overly informative error messages to users.
* For upgraded systems, latest security features are disabled or not configured securely.
* The security settings in the application servers, application frameworks (e.g. Struts, Spring, ASP.NET), libraries, databases, etc. not set to secure values.
* The server does not send security headers or directives or they are not set to secure values.
  + Modern browser headers
  + Packet called helmet
* The software is out of date or vulnerable (see [A9:2017-Using Components with Known Vulnerabilities](https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A9-Using_Components_with_Known_Vulnerabilities)).
  + We have typically seen warnings on our GitHub repos last year when we used some of our start-code containing outdated dependencies.

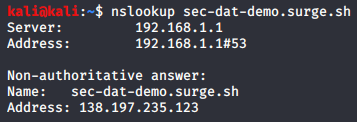
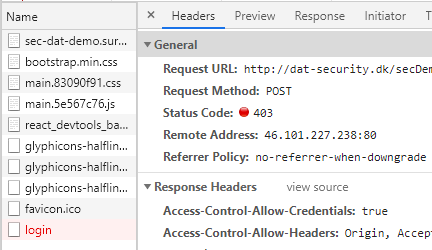
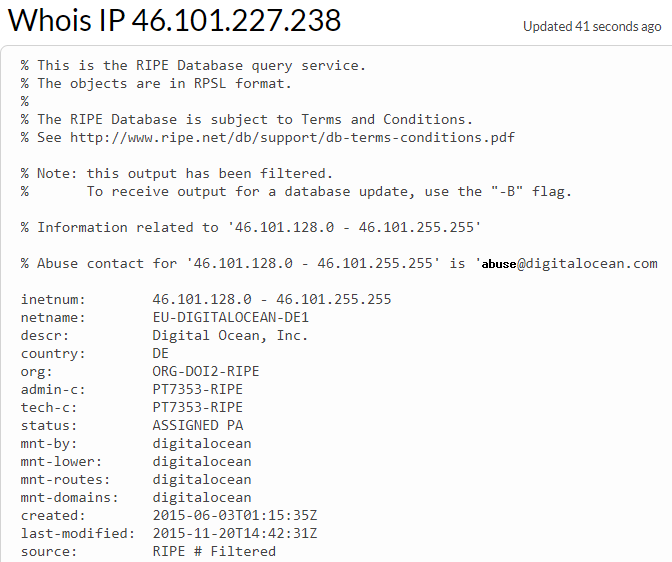
1 A6, Using default settings

This [link points](http://sec-dat-demo.surge.sh/) to an extremely simple Single Page Application, providing a login page and, after a successful login, a page using a single protected REST-endpoint.

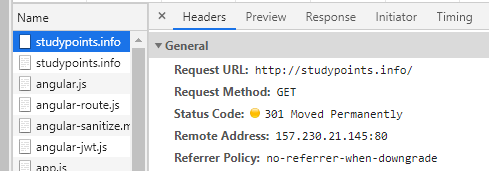
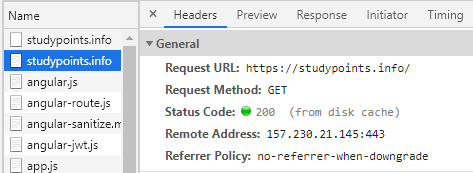
**a)** See whether you can discover the following properties of the application (not all are necessary security-problems). Use the GUI provided by the application (as a start), Postman, nmap and obviously your browser's Developer Tools, when probing the app:

* OS
  + Vi kender domæne navnet og vi kan bruge nslookup til at finde IP’  
      
    Eller i browseren:  
    
  + Responce header i browseren sider at den køre. Linux ubuntu  
    
* Server Architecture (Come up with a “guess” and provide arguments for your suggestion)
  + Lavede et forkert login og fik en tomcat fejl – Tomcat
  + Ved at scanne med nmap på backenden 46.101.227.238 kan vi se de porte derer åbne på serveren.  
    
  + Af dem er port 80 interessant da der her ligger noget vi kan scanne.
  + Vi kan med nikto se at serveren køre nginx 1.14 ubuntu på port 80  
    
  + Der ligger https på port 443, nginx reverce proxy
  + På port 3306 køre der mysql
* Server(s)
  + Ubuntu
    - Nginx
    - Mysql
* Programming Language
  + Frontent er lavet i react som en spa – Javascript – man kan ikke skjule noget javascipt  
    
  + Backend er lavet i Java  
    
  + Den fejl vi fik i browseren kom fra en GlassFish pakke.
* Important packages, classes used by the Programming Language
  + 
* Can you see “what kind of pages” logged-in users will see, without having a way to log in?
  + Vi kan se kigge i javascript på clienten of finde noget react router.
* Can you discover the client technologies used
  + React -> node\_moduels
* Default users and Passwords = the ability to login
  + Der blev brugt en default user med  user,test
* 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?
  + Vi kunne se at der bruges JWT til at store lokalt i klienten  
      
    Ved at logge ind med den default user user, test som vi fandt kan vi se hvor denne token bliver gemt.  
    
  + Ved at decode vores token kan vi se hvor lang tid den der valid  
    Vi kan tage denne token fra local storrage, og copy paste den ind på siden jew.io der kan decode den fra base64. Hvis der er en hacker der får fat i denne token, kan personen bruge den ind til den timer ud. Han behøver ikke engang at bruge en klient, men kan i stedet bare bruge postman for eksempel. Man kan dog ikke ændre i denne token da serveren også holder styr på at det er den rigtige token gennem en HS256 hash algoritme som den sammenligner på.  
    
* How/where is the token stored by the client
  + The token is stored on the browsers local storage
* Can you determine/guess(must be qualified) whether front-end, REST back-end and Database is running on the same or on different servers?

*You are hereby granted permission to scan the server hosting the BACKEND*

* Can you determine which database is used by the backend?
  + Hvis vi burger nmap kan vi se at der på port 3306 køre en mysql server. Denne port burde være lukket for udefrakommende.
* Have you discovered any unnecessary features which are enabled or installed (e.g. unnecessary ports, services, pages, accounts, or privileges)
* Who owns the domain used for the server?
  + Ved at bruge nslookup kan vi se hvem der ejer domænet og hvad IP-en er.  
    
  + Ved at skrive domænet fra front enden ind kan vi finde front end IP’en som er 138.197.235.123
  + Hvis vi prøvet at logge ind kan vi se red responce header der kommer ra backend serveren og derved finde IP’en til vores backend som er 46.101.227.238 .  
    
* Is the server hosted “privately”, by a cloud provider, or …..?
  + Hvis vi googler whois lookup, kan vi finde ud af hvem der ejer serveren og hvem der hoster den.  
    
* … Can you detect/discover more properties of the application than those suggested above?

       Extra (leading to the next exercise):

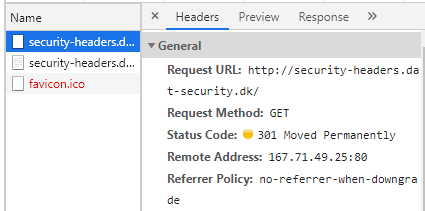
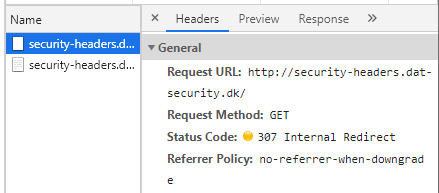
* Open developer tools, and the network-tab. Enter this URL (**exactly** as given) [http://studypoints.info](https://studypoints.info/)
  + We start by trying to access through http, and we are then redirected to https.  
    
  + Det er engine der sørger for at redriecte os fra http til https.  
    
  + Overgangen mellem redirectet fra http til https kan dog blive

Explain the first two requests, you monitor. Is this a problem, could this have been done better” (this probably require that you have read the suggested readings related to security-headers)

HSTS addresses the following threats:

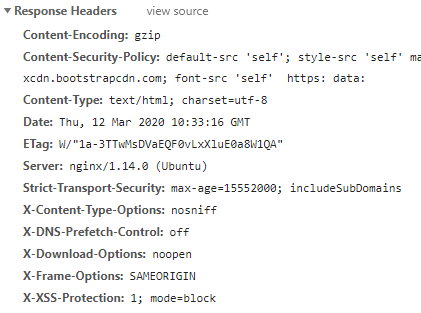
* User bookmarks or manually types <http://example.com>  and is subject to a man-in-the-middle attacker
  + HSTS automatically redirects HTTP requests to HTTPS for the target domain
* Web application that is intended to be purely HTTPS inadvertently contains HTTP links or serves content over HTTP
  + HSTS automatically redirects HTTP requests to HTTPS for the target domain
* A man-in-the-middle attacker attempts to intercept traffic from a victim user using an invalid certificate and hopes the user will accept the bad certificate
  + HSTS does not allow a user to override the invalid certificate message

Når man er blevet registered som at have besøgt en hjemmeside som https en gang kan vil man altid blive redirected til https siden frem for nogen http side.

Exercise 2 Security Headers  
Hvis man tilgår siden første gang får men en 301 moved permanently redirect fra http til https. Ved dette meget korte besøg på http han der nå at ske en masse snavs for hakcerne at misbruge.  
  
Hvis vi tilgår siden anden gang for vi en kode 307 internal redirect, hvor browseren overhoved ikke når at tilgå http, men bare videresendes til https internt i browseren. Man får her ikke lov til at være kortvarigt på http, men sendes med det samme til https.  


The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a <frame>, <iframe> or <object>.

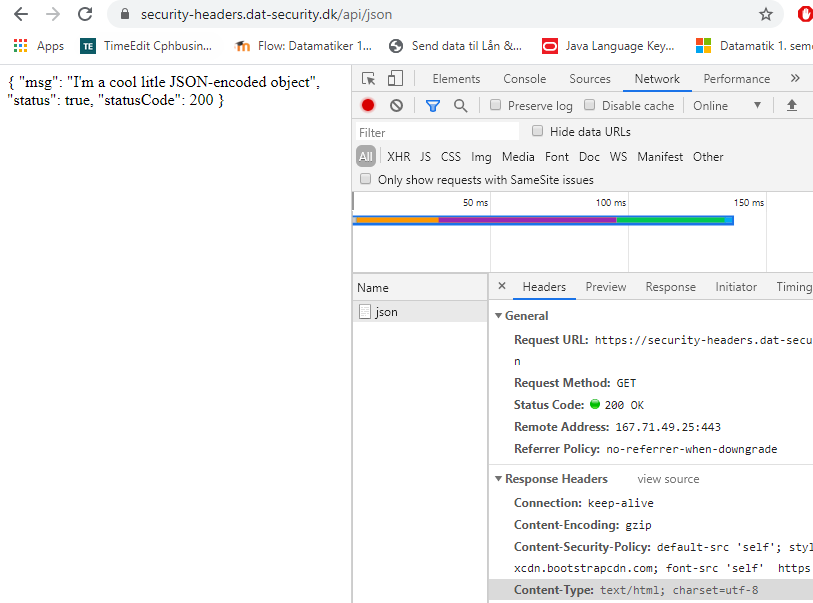
Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.  
  
The added security is only provided if the user accessing the document is using a browser supporting X-Frame-Options.

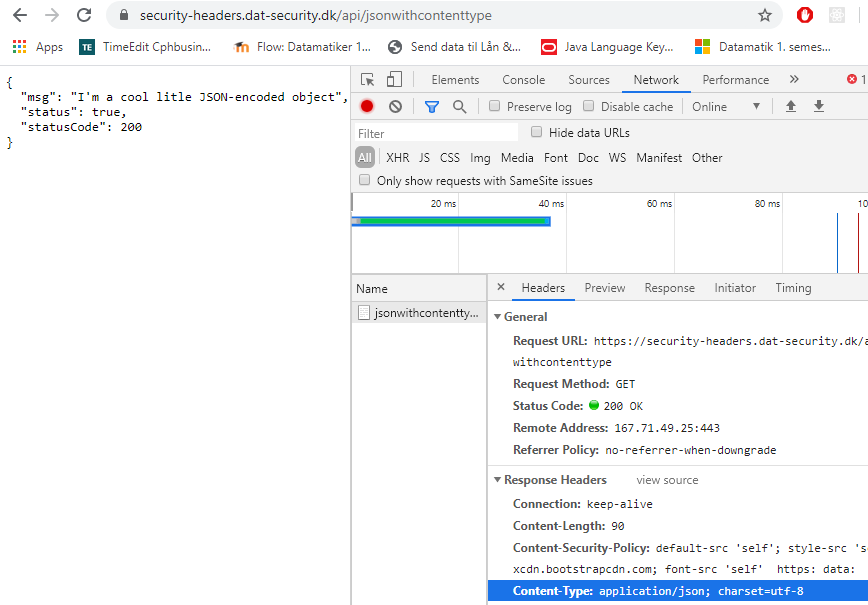
X-Frame-Options kunne tidligere misbruges af hackere til at plasere et skjult vindue for at snyde brugeren til at tro at de navigere en anden side en de faktisk gør.  


x-Frame-Options: SAMEORIGIN – sikre at man kun kan bruge en x-frame hvis man er fra samme server.

A nifty attack known as **MIME type confusion** was the reason this header was created. Most of the browsers employ a technique called MIME sniffing, that consists on taking an educated guess at what the content type of the server response is, instead of trusting what the headers content type value says.

Under certain circumstances, browsers can be tricked into making the incorrect decision, allowing attackers to execute malicious code on victim’s browsers. +

Browsere fortolker …  




En header hvor man kan finetune hvad klienten egentlig må.