1. Forbedre systemet med kunnskapen vi har tilegnet oss så langt i kurset, samt tette sikkerhetshull som kommer frem i rapport fra “hackergruppe”

- Inputvalidering (Både SQL og HTML/PHP injections)
    - Bilde Opplastning (Stian ser på dette) Bilde opplastning har nå et regulær uttrykk som sjekker om filnavnet ender med png eller jpg. Hvis det ikke har det vil man få en feilmelding som sier “INVALID FILE”. Har også lagt til en maks filstørrelse på 5MB hvor grensen på filstørrelsen ikke er nevnt noe sted da det gjør det enklere å evt utføre DOS med det (Kan diskuteres om vi skal ha med hvor stor filen kan være eller ei. Merk det er en feil med feilmeldingen som kommer opp da sjekken fungerer, men jeg får feilmeldingen som kommer når man ikke oppgir navn. Har ingen anelse hvorfor.)
    - Kommentarer og svar (Se regulærutrykk på “process-stud/fore-reg.php”. Den ser etter de fleste spesialtegn så kan man vurdere selv om alle skal bli utelukket eller ei)
    - Innlogging
    - Pinkode (tarjei)
    - Oppretting bruker(?) (Stian) Har lagt inn inputvalidering på navn, studiekull og studieretning på student registrering og inputvalidering på navn på foreleser registrering.
- API
- Innstillinger Apache server(Martin) (sette opp egen config for hver “site”?)
- Logging(!) (Alle bør skrive inn en kommentar der de selv mener det er lurt å ha logging på ting de selv har jobbet med. (Trenger ikke å ha med fare nivå.)) (Logging klar til bruk. Må bare implementeres der det gir mening.)
- Database (Brukere, rettigheter)

2. Risk management framework(excel) (Mest finpuss som må til her)
3. Sikkerhetskrav og abuse cases (^)
4. Code review(verktøy)
5. Risk-based security test (basert på krav, abuse case mm) - kan gjøres manuell eller med verktøy
6. Rapport - hvilke endringer vi gjør, og hvorfor
7. Kildekode og konfigurasjonsfiler skal ligge på “dokumentasjonssiden”

- **Inputvalidering**
    - **Prepared statements - sql injections**
    - **Regulæruttrykk**
    - **Filopplasting - whitelist, kun png og jpg og størrelse**
- **HTTPS**
    - **Endra her i 000-default.conf og sertifikat-site.conf**

- **Endra innstillinger i apache2.conf**
- **Lagt inn logging via Graylog**
    - **Logger innlogging**
    - **Bruk av “farlige” tegn i felter med bruker input**
    - **Logger opplastning av filer**
- **Database - lesebruker, skrivebruker**

Risiko analyse: *Risk Management Framework*

	Scenario	Sannsynlighet	Risiko	Score	Abuse Case	Sikkerhetskrav	Status	Ansvar
2	xss på kommentarer/meldinger	4	4	16	en angriper legger inn et script som kommentar i inputfeltet.	Inputvalidering		
3	Tilgang til SELECT og INSERT statements i API	3	5	15	En angriper benytter SQL-spørringer for å få tilgang til data som brukeren ikke skal ha tilgang til	Ha prepared statements		
4	Opplastning av andre filer enn jpg	5	5	25	Angriper kommer seg rundt filvalidering og kan laste opp skadelig kode	Whitelist/blacklist		
5	Stjele cookies via XSS	2	1	2	en angriper får tilgang til en annen brukers innlogget sesjon	Inputvalidering		
6	Denial of service angrep	3	2	6	en angriper utnytter manglende begrensninger på antall forespørsler og fyller opp databasen med data			
7	Validere EXIF data	2	3	6	en angriper endrer exif data på et bilde slik at det inneholder php kode			
8	Login informasjon direkte lagt inn i php filer	1	4	4	En angriper oppnår å kunne lese av php filer i systemet og kan dermed se login informasjon til databasen.			
9	SQL injection	3	5	15	En angriper benytter SQL-spørringer for å få tilgang til data som brukeren ikke skal ha tilgang til	Inputvalidering		
10	Alle handlinger blir utført av root bruker (en rolle)	3	3	9	en angriper kan få tilgang til root brukeren og får alle rettigheter	Flere brukere/rettigheter		
11	At GIT ikke er privat	2	5	10	en angriper finner git reposit og kan se sensitiv systeminformasjon	Privat GIT		
12	Det utføres ikke sikkerhetsoppdateringer	2	3	6	En angriper finner et sikkerhetshull som gir dem tilgang til systemet gjennom en bakdør			
13	HTTP	3	3	9	En angriper kan se dataen i som blir sendt gjennom nettsiden ved hjelp av programmer som wireshark.	HTTPS		
14	Webserver			0				
15								

RISIKOBILDET						
		Svært lite sann	Lite sannsynlig	Sannsynlig	Ganske sannsy	Svært Sannsynlig
	Sannsynlighet,	1	2	3	4	5
Katastrofal	5	5	10	15	20	25
Svært stor	4	4	8	12	16	20
Stor	3	3	6	9	12	15
Middels	2	2	4	6	8	10
Liten	1	1	2	3	4	5

Grønt område = Akseptabelt

Gult område = Vær oppmerksom

Rødt område = Bør absolutt gjøres noe med, skadelig høy risiko

Endringer:

Vi har gjort kode reveiw manuelt og fant ut at det var en del ting i koden og strukturen vi måtte endre.

Vi har lagt til logging på de fleste stedene der systemet kræsjer pga brukerfeil. greylog

Vi har endret spøringer til prepared statements.

Vi har lagt til input validering på input felter. Dette er gjort med regulæruttrykk .

Vi har endret api til å ha prepared statements og inputvalidering og logging.

Pin kodene til fagene er nå skjult i koden og hentes fra databasen.

Lagt to filer som holder på sensitiv informasjon (passord/navn etc.) til databasen i /var/www/private slik at dette ikke skal kunne bli sett. Disse blir henvist til to databasefiler i /html.

Lagt til regulær uttrykk på de fleste input-felter.

Endret fra http til https.

Innlogging har vi endret til prepared statements.

Bildeopplasting- Har lagt til inputvalidering som bare tillater .png og .jpg filer. Har også satt en maksgrense på størrelse på filene som blir lastet opp.

Apache – lagt til/endret diverse innstillinger i konfig-filer; apache2.conf, 000-default.conf, sertifikat-site.conf.

Db brukere- la til bruker for lesing og skriving til databasen.

Endret passordet til root.

Vi la til tilkoblingen til databasen i en privat mappe.

Endringer vi ikke fikk gjort:

Vi fikk ikke gjort metodene i api til _POST fra _GET. Vi prøvde.

Endre pinkodene til noe annet enn det samme som var i steg 1, så de har essensielt kodene fra før.

Endre alle filene til å bruke forskjellige roller til databasen. (skrive og lese brukere) vi har de forskjellige brukerne men hadde ikke tid til å endre alene filene siden det hadde vært mye å endre med tanke på at mange av filene skal lese og skrive til db og det ville blitt krøll siden de bruker mange av de samme variablene.