Enkel risikoanalyse av arkitekten

(Enkelte mangler her som grupperinger av systemet)

1. Vi vil ikke at kode skal kunne bli lastet opp via input-felter eller fil opplasting.

2. Dette kan skje via en svakhet i systemets kode, eks. med hjelp av SQL-injections eller ved fil opplasting hvor en hacker laster opp ondsinnet programkode som utfører visse handlinger/tar kontroll over siden.
3. Dette er hindret med hjelp av prepared statements, input validering på tekstfelter og ved fil opplasting at det bare er lov med .png/.jpg filer. (API har også prep state/input val).

Stor sannsynlighet for at et slikt angrep blir utført, og avhengig av hva koden inneholder kan konsekvenser rangere fra "tull" til alvorlig. Selv om det er begrenset sensitiv informasjon man kan få fra brukerne i dette systemet, kan angrep som dette føre til for eksempel malware som blir spredt som åpner for mye mer. Høy risiko.

1. Vi vil ikke at noen skal kunne få kontroll over hele systemet.

2. Dette kan f.eks. skje om en hacker klarer å få tilgang til root user, som vil gi full kontroll. Kanskje til og med farligere her er ansatte og andre som har jobbet i bedriften.
3. Vi har laget en bruker for lesing og en for skriving i databasen.
4/5. Bare ha en/(få!) personer som har tilgang til root user. Lag andre brukere som får sine rettigheter satt basert på hvilke handlinger de skal kunne utføre.

Her trengs det et spredt spenn – kan for eksempel være sikkerhetshull gjennom nettsiden eller databasen som blir utnyttet, brute-force angrep, "insider"-folk (enten gjennom at insider gjør angrepet selv eller at de blir lurt av en hacker), social engineering osv. Så det er vanskelig å gi en nøyaktig sannsynlighet for at det skjer, men vi mener middels, og risikoen er fatal med tanke på at nå kan angriperen gjøre hva den vil.

1. Vi vil ikke at brukere skal få frastjålet/bli låst ute av sin konto.

2. Her er det mange muligheter, men kanskje viktigst er å tenke på social engineering. Nå er det vanskelig å ha tiltak mot dette, annet enn sunn fornuft fra brukeren selv, men ekstra lag med autentisering kan hjelpe til for å forhindre slike angrep i en hvis grad (avhengig av situasjonen).
3. Nei.
4/5. Som nevnt over burde systemer som 2faktor-autentisering blitt brukt, eller i andre tilfeller som f.eks. mot noe ala brute-force angrep hvor vi har en sperre på antall forsøk på x tid.

Her kan det diskuteres om det er liten eller stor sannsynlighet (med tanke på at vi ikke har noen reell autentisering på plass), siden per nå er nettstedet primært et meldingssystem mellom elev og lærer, og bortsett fra registreringsinformasjon er det ikke mye å hente her (så lenge meldinger ikke inneholder hemmeligheter). I verste fall blir eleven/foreleser uten en konto en x-tid til det er ordnet opp igjen (så langt man ikke har samme innlogging andre steder og dette blir prøvd..). Lav risiko.

1. Vi vil ikke at data som går mellom brukere og siden skal kunne fanges opp av andre.

2. Dette kan skje ved bruk av HTTP, hvor data vil kunne lyttes til og fanges opp av en hacker.
3. Vi bruker HTTPS som krypterer data i stedet for at det går i klar tekst.

Henviser nederst til frastjålet konto, men siden alt av data blir sendt ukryptert velger vi å sette middels (stor siden det er innlogging?) på både sannsynlighet for at et slikt angrep kan skje (med tanke på hvor lett det er å utføre) og risiko.

1. Vi vil ikke at systemet blir utsatt for angrep som skjer via sikkerhetshull i deler av systemer/programvare som blir brukt.

2. Dette kan skje om det ikke utføres regelmessige oppdateringer.
3. «Nei». Men vi har bare installert det som er blitt nødvendig (så langt vi vet).
4. Ha oppdateringer som blir gjort automatisk, eventuelt manuelt.
5. Med tanke på at oppdateringer i et system ofte kan «ødelegge» for et annet blir oppdateringer «litt nedprioritert». F.eks. enkelte versjoner med Graylog og Mongodb skaper kluss for hverandre.
Liten sannsynlighet for at et slikt angrep skjer, men konsekvensene kan være store avhengig av hva et potensielt sikkerhetshull gir til angriperen. Lav(middels?) risiko.

1. Vi vil ikke at kildekoden for PHP er/blir kjent.

2. Dette kan skje ved dårlig oppbevaring – f.eks. hacker kan få tilgang til .git stilen som er blitt lastet opp til server.
3. Vi har php-filene med innloggingsinformasjon lagret et separat sted på serveren som Apache ikke har tilgang til.