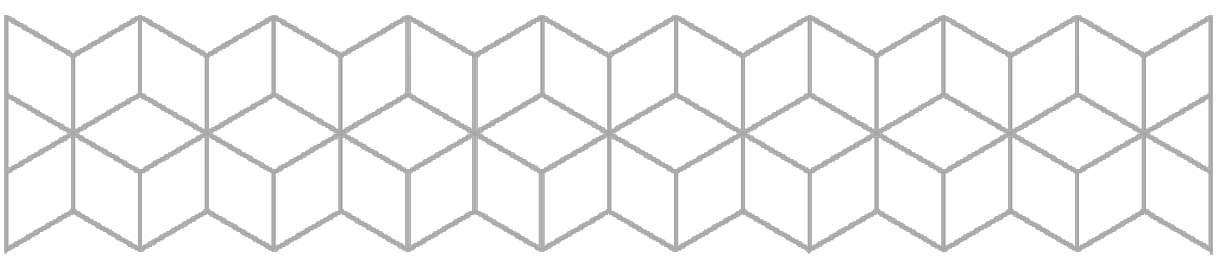
Høgskolen i Østfold

SENSORVEILEDNING

Emnekode:	ITF25019
Emnenavn:	**Datasikkerhet i utvikling og drift**
Eksamensform:	**Skoleeksamen**
Dato:	25/05-2022
Faglærer(e):	Tom Heine Nätt
Eventuelt:	



Info

Generelt skal karakteren settes basert på et helhetsinntrykk/helhetsvurdering av besvarelsen, ikke direkte på poenggivning i oppgaver og grenseverdier på en totalscore.

Det er en del av den enkelte sensors mandat å vurdere hvordan uttelling skal gis på svarene i de ulike oppgavene, og hva som er et akseptabelt nivå. Dette vil også være en del av diskusjonen mellom sensorene ved sensureringen.

Resten av denne sensorveiledningen gir noen stikkord til momenter som bør være med i et godt svar (enkeltmomenter kan erstattes av andre, da det ikke nødvendigvis er forventet at studentene kommer på nettopp disse). **Merk at sensorveiledningen ikke er en komplett liste med alle momenter eller nødvendigvis et A/B-svar**. Studentbesvarelser bør utdype momentene mer, og sette de i en sammenheng, slik at sensor kan være trygg på at teorien er forstått og ikke bare pugget/tilfeldig.

Hovedhensikten med denne sensorveiledningen er å gi sensor et innblikk i relevant pensum som kan passe i de ulike oppgavene. Det er altså ikke et løsningsforslag.

Sensorene må også se på hvordan kandidatene som en gruppe har løst eksamenssettet, og legge forventingene på hver oppgave deretter.

Dette er et fag som går i 2. og 3. studieår. Drøftinger og diskusjon, ikke kun oppramsing av pugget teori som en "fasit" bør derfor gi ekstra uttelling.

Oppgave 1

Denne oppgaven har ingen fasitsvar. Studenten bør imidlertid vise forståelse for tematikken. Selv om ikke oppgaven har noe fasitsvar, bør studenten i hvert fall berøre følgende(eller lignende momenter):

- Mitigation
- "Shift-left" ("building security in")
    - Besparelser
    - Bedre sikkerhetstiltak/-løsninger
- Risk management
- Bugs vs. flaws / code review vs. design analysis
- Konseptene om å avverge vs. å oppdage/redusere konsekvenser
- Ønsket om å lage et produkt uten feil i stedet for å fikse feil i et produkt.
- Krever endringer i arbeidsmetoder og kompetanse hos utviklere og prosjektledelse

Oppgave 2

Denne oppgaven har ikke noen absolutt liste, men det er forventet at studenten nevner:

- `Content-Security-Policy`
- `X-frame-option (evt. frame-ancestors under content-security-policy)`

- `Permission-policy (evt. Feature-policy som den erstatter)`
- `Strict-transport-security*`

*Dette kan også bli nevnt i oppgave 5a. Er det beskrevet godt nok i 5a til å kunne passe inn her (eller henvist), så kan det telle.

Andre som kan nevnes (ikke en endelig liste), men ikke er forventet:

- Referrer-policy
- X-XSS-Protection (*Kommentar: Denne er nylig blitt deprecated, men det ble ikke nevnt i forelesning og det er dermed ikke forventet t studentene har fått med seg*)

Ut over å nevne headeren, bør studenten argumentere med hvorfor den bør benyttes og hvilke muligheter den gir.

Oppgave 3

a) Her bør i hvert fall følgende nevnes og beskrives:

- SQL-Injection og command-injection
- XSS
- DoS-angrep (som går på ressurser)
- DoS-angrep (som går på feilverdier)

Ekstra uttelling til de studentene som nevner at i utgangspunktet foregår alle angrep gjennom en form for input.

b) Mange mulige løsninger, en er /[0-3]\d\-\[0-1]\d\-[1-2]\d\d\d/

Også /\d{2}\-\d{2}\-\d{4} og lignende bør gi god uttelling, da de er riktige, selv om det muligens burde vært en enkel sjekk på lovlige verdier på hver posisjon. Hovedhensikten med oppgaven er å se at kandidaten forstår regulæruttrykk.

Oppgave 4

Her bør muligheter i rettighetssystemet beskrives med spesifikke handlinger (SELECT, INSERT osv) på database, tabell og kolonne.

I tillegg bør kandidaten beskrive:

- Rettigheter sammen med views (overordnet hvordan det gjøres, samt muligheter det gir)
- Rettigheter sammen med prosedyrer (overordnet hvordan det gjøres, samt muligheter det gir)

Det er positivt om kandidaten beskriver hvordan flere ulike DB-brukere i ulike deler av en applikasjon kan inndele rettigheter mer.

Det er positivt om kandidaten beskriver hvordan "host"-delen av DB-brukere kan være til nytte med tanke på rettigheter.

Oppgave 5

a) Her bør kandidaten ha beskrevet:

1) Anskaffelse av (generering av) sertifikater
2) Overordnet (ikke teknisk) at man må installere/konfigurere sertifikatene i webserver og konfigurere webområdet til å benytte SSL
3) Redirect av HTTP-requests til HTTPS-requests.

I punkt tre er det ekstra uttelling for de som diskuterer kort forskjellen på en relativ redirect (samme område i HTTPS) og en absolutt redirect (videresendes til forsiden). Også god ekstra uttelling for de som diskuterer HSTS som en del av steg 3

b) Kandidaten bør nevne at dette er en filtrering på applikasjonslaget (HTTP/HTTPS) og at den er spesielt egnet egnet for å oppdage og stoppe skadelige web-forespørsler. Dette kan være XSS- og Dos-angrep, SQL-injection osv. En normal brannmur fokuserer i motsetning til en WAF på nettverkslaget og transportlaget.

Oppgave 6

Studenten bør som minimum nevne at vi med en if-test kun kan fange feil som vi vet at kan skje og delvis hva skyldes. Med en exception kan vi også fange ukjente feil.

Det teller positivt om studenten diskuterer global exception handlers, noe det ikke finnes tilsvarende "if-test" for.

Oppgave 7

a) Dersom tjenestens standardverdier endres, vil vi beholde de verdiene vi selv har tatt et bevisst valg for/på. I tillegg tvinges vi til å vurdere om standardverdiene er vår beste verdi, og ikke bare blir vår verdi fordi vi ikke gidder/tenker på å finne noe annet. Ulempen med egne verdier kan være at endrede standardverdier fra tjenesten har en viktig hensikt (vi må følge med på hva som endres)

b) Standardverdier er laget dels for å dekke flest mulig bruk av systemet og dels for å la brukere komme raskt i gang og ikke få unødvendige problemer. Det er derfor sjelden standardverdier er optimale innen sikkerhet. Det vil også være umulig å lage standardverdier som er "optimale" for alle tjenester, da innstillinger også ofte har bi-effekter som må vektes opp mot trusselbildet (ulike sikkerhetsbehov og trusselbilder)

c) Her bør det nevnes at endringer i konfigurasjon kan være en rask måte å skru på sikkerhetsmekansimer eller innføre begrensinger (mye raskere enn å kode om systemer) under angrep. For eksempel kan studenten nevne at ved DoS-angrep kan man senke timeout eller begrense systemressurser pr. forespørsel. Et annet alternativ er at man enkelt kan stenge ned deler av et webområde.

En viktig ide er at det er en fordel å ha alternativ konfigurasjon for ulike hendelser klar på forhånd. Gjerne som alternative konfigurasjonsfiler, en i hvert fall som dokumenterte mulige endringer.

Oppgave 8

a) Tekniske: DoS pga store mengder forespørsler. Organisatoriske: Tømme registre for kritisk/verdifull data, overvåke tjenester (lagerstatus, ledige timer etc.)
b) Mulige teknikker å nevne (ikke en fullstendig liste):
    a. Ulike captchas (kan aktiveres ved mange forespørsler)
    b. Tidsdelay
    c. Avtaletekster/advarsler (ikke teknisk)
    d. Knytte oppslag til brukerkontoer (blokkere ved for mange)
    e. Knytte oppslag til bruker gjennom cookies (blokkere ved for mange)
    f. Hyppige endringer i nettsidens struktur
    g. Unngå gjettbare url-er (nummererte indekser med mer)
    h. Laste informasjonen via JS eller lignende
    i. Blokkere etter for mange "ugyldige oppslag"

Oppgave 9

a) Token er en "hemmelighet" som er delt ut av tjenesten til brukeren og som kan gi aksess til tjenester eller funksjoner. I mange tilfeller kan det også identifisere brukeren. Som regel håndteres token av nettleser/applikasjon, og er ikke noe brukeren direkte ser. Kan bl.a. benyttes til:
    a. Validering av at brukere har vært innom skjemaer før innsending (for eksempel motvirke CSRF)
    b. Som identifisering og aksess til API
    c. Som en del av multifaktor autentisering
b) Tokenet inneholder all nødvendig informasjon slik som utløpsdato, bruker osv. Dermed slipper man vedlikeholde denne informasjonen på server. Informasjonen er "signert" med en secret key i en sjekksum, slik at informasjonen ikke kan endres hos klient.

Oppgave 10

Bør vurdere:

- App må identifiseres ved et token/sertifikat
- Identifisering av server
- Kommunisere via HTTPS
- Inputvalidering av forespørsler til server (og svar til klient)
- Sjekksummer
- Tidsstempler
- Sekvensnumre

Noen prinsipper:

- "App kan ikke stole på server, server kan ikke stole på app"
- All kommunikasjon må ta høyde for manipulering, og API kan sidestilles med et GUI i client/server-modellen.
- Utføre testing av API (både gyldige og ugyldige handlinger/verdier)
- Begrense API til det nødvendige