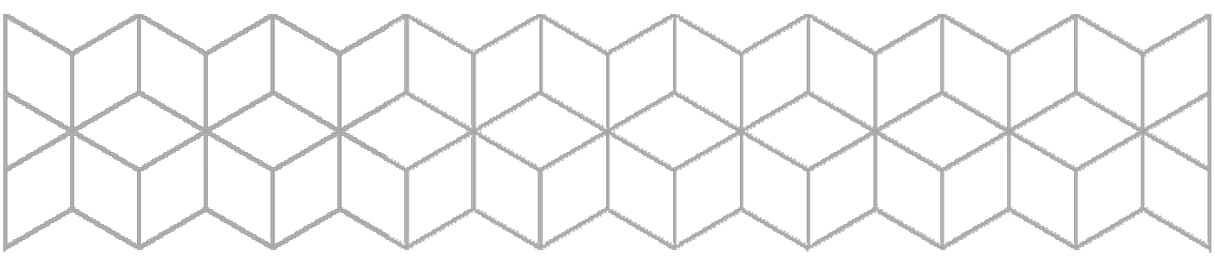
Høgskolen i Østfold

SENSORVEILEDNING

Emnekode:	ITF25019
Emnenavn:	**Datasikkerhet i utvikling og drift**
Eksamensform:	**Skoleeksamen**
Dato:	24/05-2023
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):

- Begrepene bør defineres
    - Kortversjonen er at sikkerhetsarbeidet flyttes tidligere i utviklingsprosessen (mot venstre), slik at man ikke lager noe, og så tester det. I stedet integreres sikkerhetsarbeidet i kravspec, design og implementasjon. Testingen til slutt er nå heller primært for å se at sikkerhetsarbeidet har vært godt nok, ikke avsløre feil.
- Fordeler:
    - Kostnadsreduserende (langt billigere å «Sikkerhetsteste» og rette design enn et ferdig produkt
    - Bedre løsninger. Å løse sikkerhetsutfordringer på et ferdig produkt gir sjelden gode løsninger men mer «plasterlapper». Når man kan endre designet kan man få gode løsninger.
- Utfordringer:
    - Krever mer sikkerhetskunnskap av alle prosjektdeltakere, ikke bare av en gruppe som tester det ferdige produktet
    - Krever at sikkerhet er et prioritert område.

Oppgave 2

a) Hensikten er primært å samle loggdata fra flere ulike delsystemer. Gevinsten vil være at vi enklere kan analysere og følge hendelser på tvers av komponenter. I en sikkerhetssamenheg vil det kanskje være «totalen» som indikerer en hendelse, ikke hver handling i de ulike systemene. I tillegge vil man kunne benytte spesialiserte og effektive verktøy til prosessering, analyse og varsling.

b) Følgende metoder er vanlige (finnes flere og må ikke nødvendigvis deles inn på samme måte):

- **a.** Hendelser gjort av brukerne eller systemet (typisk innlogging):
- **b.** Spesielle situasjoner vi kan detektere med tester (typisk inputvalideringsfeil)
- **c.** Spesielle hendelser som kan detekteres med unntak (exceptions) (typisk skrivefeil til disk)
- **d.** Bruksmønstre og sammensatte handlinger (typisk brukeren skrev feil passord fem ganger)

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.

Oppgave 3

a) CSP kan settes som headere eller meta-tagger og gir nettleseren beskjed om hvordan og hvor fra eksterne ressurser i nettsiden skal kunne lastes. I tillegg kan det sette begrensinger på hva eksternt innhold skal kunne gjøre av handlinger.
   a. Typisk kan vi for eksempel si at script-kode kun skal lastes via HTTPS fra eget domene, og at inline og eval ikke er tillatt.
   b. Vi kan også som et eksempel liste opp hvilke domener ut over vårt eget som bilder kan lastes fra
   c. Det kan defineres hvilke nettsider som det er tillatt å «automatisk» navigere til eller sende skjemadata til

b) CSP beskytter brukeren gjennom at nettleseren til brukeren iverksetter begrensinger. Typisk kan SOP begrense mange XSS-angrep der brukeren blir vist innhold som forsøker gjøre handlinger som ikke samsvarer med hva som er satt som tillatt i SOP-headeren,. Det er altså ikke en teknikk som hindrer angrep mot server/tjeneste direkte, men indirekte gjennom angrep på våre brukere.

c) Sandbox vil når det tas i bruk fjerne alle rettigheter innholdet har, for så å whiteliste enkeltrettigheter slik som å tillate skjemaer, scripts, popup osv. Benyttet sammen med iframes betyr det at vi kan velge hva innhold (som potensielt kan forandre seg over tid) får lov til å gjøre mot våre brukere.

Oppgave 4

a) Kan skilles på:
   a. Inndeling 1:
      i. Whilelisting: Forteller hva som er tillatt

ii. Blacklisting: Forteller hva som ikke er tillatt

b. Inndleing 2:
   i. Synatktisk: Mønstre og struktur
   ii. Semantisk: Meningen i innholdet
      1. Semantisk riktig: F.eks. antall varer må være større enn 0, bokstavkombinasjonen på bilskiltet er blant de som benyttes
      2. Gyldig: E-postadressen er i drift, bilskiltet er registrert

c. Inndeling 3:
   i. Klientside; For brukervennlighet
   ii. Serverside: For sikkerhet

d. Metoder:
   i. Lengde
   ii. Mønster
   iii. Tegn
   iv. Unikhet
   v. +++

b) Ettersom inputvalidering er en ganske konkret sikkerhetsmekanisme som er enkel å teste, kan det være nyttig å heller fokusere på at system skal være sikkert uten inputvalideringen. Når vi si at det er det, vil vi oppnå «mitigation»/»defence-in-depth» når vi så legger på inputvalideringens om et siste lag med sikkerhet.

c) F.eks. /\d{2,5}\-[A-Z]{2}(\.[A-Z])?/

Oppgave 5

a) Herding betegner prosessen med å gjøre enheter og tjenester sikrere ved å endre konfigurasjonen eller fjerne funksjonalitet. Som regel utføres endringene som et ekstra sikkerhetstiltak, uten å kobles til noen bestemte sikkerhetstrusler. Med andre ord som en del av mitigation og defence in depth. Typiske eksmepler kan være:
   a. Fjerne avslørende informasjon om versjoner osv.
   b. Begrense ressursbruk for forespørsler til det nødvendige
   c. Skru av funksjonalitet som ikke benyttes (men som kan innebære sikekrhetshull)
   d. Fjerne støtte for eldre protokoller, standardbrukere, eksempeldata osv.
   e. Endre nettadresser, brukernavn, lagringsposisjoner osv. bort fra standardverdier.

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)

Oppgave 6

a) Tekniske: DoS pga store mengder forespørsler. Organisatoriske: Tømme registre for kritisk/verdifull data, overvåke tjenester (lagerstatus, ledige timer etc.). Skraping glemmes ofte fordi handlingene skraping består av er ønskede handlinger fra våre brukere. Det er bare «for mye bruk» som vi vil unngå. Derfor er det litt uklart hva som er «for mye»,

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"

c) Både skraping og de DoS-angrepene som baserer seg på overbelastning er begge forårsaket av for mange forsøk/forespørsler som går ut over normalt bruk. Derfor vil mottiltakene ofte være sammenfallene (captchas, blokkering, limits for brukerkontoer osv).
Det er imidlertid viktig at studenten nevner denne avgrensningen blant alle typer DoS-angrep i sitt svar

Oppgave 7

a) Det er helt vesentlig at som minimum hver applikasjon har en egen databasebruker der rettigheter er begrenset/tilpasset til hva applikasjonen skal kunne gjøre. Består applikasjonen av ulike deler/roller (typisk foreleser/student) så kan det være fordelaktig med brukere til de ulike delene.
b) 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 "host"-delen av DB-brukere kan være til nytte med tanke på rettigheter.