OWASP ble opprettet i 2001. Ikke-kommersiell, internasjonal organisasjon. Laget lista over 10 vanligste sårbarheter.

1 A1 - injection

Når en applikasjon kan ta imot ikke-validert brukerinput. Feks SQL-injection. **Løsning:** Bruk et trygt API som paramatriserer input før spørring. Bruk prepared statement. Validere alle input på siden.

2 A2 - Broken Authentication and Session Management

- Passord lagres i klartekst uten bruk av hashing
- PAssord kan overskrives gjennom svak konto-administrasjon
- Sesjonsid eksponeres i URL
- Passord og sesjons id sendes ukryptert

Løsning: Ikke bygge egne rammeverk for autentisering og sesjonshåndtering.

3 A3 Cross-Site scripting (XSS)

Vanligste sårbarheten i web apper. Oppstår når en applikasjon viser frem data gitta av en bruker i en webside uten at dataen er validert(escapet). Angripere kan få mulighet til å se skript i en nettleser for å ta over sesjoner og omdirigere. **Løsning:** Escape brukerinput. Validere brukerinput. Ikke tillat avansert input. Validering fikser alle problemer med XSS for vanlig data.

4 A4 Insecure Direct Object References

Verifisere at bruker har tilgang til objektet i URLen. Feks. kontoid=123 i urlen bør kun kunne nås av bruker nr 123.
Løsning: Bruk indirekte objektreferanse. I stedet for å bruke databasenøkkel direkte, kan man bruke en indirekte referanse som kun gjelder sesjonen. Man kan lage en liste over ressurser den påloggede brukeren har tilgang til i sesjonen, og kun bruke indeksen som nøkkel i referansesne. Bruke mapping.

Sjekk aksess. For hver bruker av en objektrefreanse, sjekk om brukeren er autorisert for objektet.

5 A5 Security Misconfiguration

Feilkonfigurasjon kan skje mange steder i stacken. Er noe av programvaren utdatert? **Løsning:**

- Automatiser installasjon av nye miljøer for å fjerne muligheten for manuelle feil.
- Ha god prosess som sikrer at man bruker ny programvarenHa god applikasjonsarkitektur som skiller tydelig
- Vurder periodiske sikkerthetsgjennomganger

6 A6 Sensetive Data Exposure

Ikke kryptert sensitiv data. Hvis man feks har svake nøkler. **Løsning:**

- Krypter all sensitiv data som lagres over tid i databaser eller loggfiler.
- Bruk SSL(krypter alt som sendes ut)
- Ikke ha sensetiv data i url
- Ikke lagre sensitiv data du ikke trenger
- Deaktiver autocomplete

7 A7 Missing Function Level Access Control

Applikasjoner beskytter ikek alltid funskjonene i systemet ordentlig. Feks om UI har linker til uautoriserte funksjoner. Om det mangler API på serversiden for visse funksjoner?

Løsning:

- Lag mekaniskem for autoriserbar konfigurerbar, og lett og oppdatere,
- Implementere sjekker i business logic

8 A8 Cross-Site Request Forgery

En app tillater en bruker å sende tilstandsendrene requester. En angriper kan da konstruere en URL som feks overfører penger fra offerets konto. Hvis offeret besøker en spesiell side mens de er pålogget kan angriperen utløse en url.

- For å hindre CSRF angrep kan man bruke en uforutsigbar CSRF token enten i bodyen på requesten eller i urlen.

9 A9 Using Components with known Vulnerabilites

Dette er et vanlig problem fordi utviklingsteam sjelden har fokus på å sikre at man bruker biblioteker. Mange vet ikke engang hvilke biblioteker man bruker. Om man bruker Maven eller feks NPM.
Løsning: Oppdatere biblioteker. Heartbeat bug i OpenSSL.

10 A10 Unvalidated Redirects and Forwards

Appenr sender ofte nettleseren videre fra en URL til en annen via redirect og forward. Hvis destinasjoner er baser på ikke validert brukerdata, vil en angriper kunne konstruere en url som sender nettleseren til en ekstern ondsinnet side. Phishing.
Løsning:

- Unngå å bruke redirect og forward
- Ikke være avhengig av brukerdata