KRIITTISIMMÄT HAAVOITTUVUUDET WEB-SOVELLUKSISSA

JA MITEN VÄLTTÄÄ NE SOVELLUSKEHITYKSESSÄ

Tietoturvaopas

2

Johdanto

Olemme vuodesta 2005 lähtien toteuttaneet yli 1000 verkkopalvelun haavoittuvuustestausta ja auditointia. Näistä syntyneeseen kokemukseen, sekä OWASP Top 10 -listaan pohjautuen olemme tässä oppaassa kuvanneet kriittisimmät web-sovellushaavoittuvuudet sekä miten niiden syntyminen voidaan estää.

Implementoimalla kontrollit oppaassa esiteltyjä haavoittuvuuksia vastaan, voidaan sovelluksen tietoturvaa merkittävästi parantaa. Kontrollien implementointi on hyvä ottaa huomioon jo sovellusta suunniteltaessa ja tietoturvavaatimuksia luotaessa.

Toivotamme haavoittuvuusvapaita verkkopalveluita!

- 2NS Haavoittuvuustutkimustiimi

Sisällysluettelo

1. Tiivistelmä

Kokemuksemme kriittisimmistä haavoittuvuuksista vastaa pitkälti OWASP Top 10 -listan näkemystä. Alla on esitelty kriittisimmät web-sovellushaavoittuvuudet. Nämä haavoittuvuudet voidaan jakaa muutamaan pääjoukkoon:

1. Syötteenkäsittelyyn liittyvät heikkoudet

2. Käyttäjien tunnistamiseen ja valtuuttamiseen liittyvät heikkoudet

3. Tiedon suojaamiseen liittyvät heikkoudet

4. Sovelluksen käyttämiin komponentteihin sekä alustaan liittyvät heikkoudet

Implementoimalla kontrollit ja ottamalla nämä haavoittuvuudet huomioon jo tietoturvavaatimuksia luodessa, voidaan sovelluksen tietoturvaa parantaa merkittävästi.

2. Kriittisimmät haavoittuvuudet

Injektiot
Autentikointi ja istunnonhallintahaavoittuvuudet
Cross Site Scripting
Turvattomat suorat objektiviittaukset
Turvattomat konfiguraatiot
Arkaluonteisen tiedon paljastuminen
Puuttuva funktiotason pääsynhallinta
Pyyntöväärennös (CSRF)
Haavoittuvien komponenttien käyttö
Validoimattomat edelleenohjaukset

(OWASP TOP 10 haavoittuvuudet, lähde OWASP)

2.1. Injektiot

Haavoittuvuuden merkitys

Injektiohaavoittuvuudet syntyvät, kun sovellus lähettää ei-luotettavaa syötettä, esimerkiksi käyttäjän syötettä, taustajärjestelmäkyselyyn. Haavoittuvuuden avulla hyökkääjä kykenee muokkaamaan taustajärjestelmään lähtevän kyselyn rakennetta. Tällöin hyökkääjä voi vaikuttaa esimerkiksi siihen, mitä tietoa tietokannasta palautetaan. Pahimmassa tapauksessa tämä voi johtaa esimerkiksi:

- Luottamuksellisuuden menettämiseen
- Autentikoinnin sekä muiden kontrollien ohittamiseen

Haavoittuvuuden välttäminen

Lisättäessä käyttäjän syötettä taustajärjestelmäkyselyihin, tulee siitä tehdä turvallista. SQL-kyselyiden osalta tämä tarkoittaa esimerkiksi parametrisoitujen kyselyiden käyttöä. Mikäli taustajärjestelmäkyselyn parametrisointi ei ole mahdollista, tulee käyttäjän syötteestä poistaa tai tehdä turvalliseksi taustajärjestelmäkyselyn käyttämät erikoismerkit.

2.2. Autentikointi- ja istunnonhallintahaavoittuvuudet

Haavoittuvuuden merkitys

Autentikointi- sekä istunnonhallintahaavoittuvuudet ovat tietoturvaongelmia, jotka mahdollistavat sovelluksessa olevan tunnistautumisen ohittamisen tai toisen käyttäjän istunnon kaappaamisen. Haavoittuvuudet mahdollistavat hyökkääjän toimia toisen käyttäjän valtuuksilla kohdepalvelussa. Tyypillisiä esimerkkejä haavoittuvuudesta ovat:

- Liian lyhyet ja helposti arvattavat istuntotunnisteet
- Uusia tunnuksia voidaan luoda ilman riittäviä valtuuksia
- Istuntotunnisteita siirretään URL:ssä
- Istuntotunnistetta ei vaihdeta valtuutustason muuttuessa (esimerkiksi sisäänkirjautuminen)
- Istuntojen elinikä on liian pitkä
- Evästeissä ei ole HttpOnly- tai Secure-asetuksia

Haavoittuvuuden välttäminen

Varmista, että sovelluksestasi ei löydy yllä mainittuja esimerkkejä. Varmista myös, ettei hyökkääjä kykene ohittamaan tunnistautumisprosessia tai kaappaamaan toisen käyttäjän istuntoa. Toteuta esimerkiksi tunnistautuminen ja istunnonhallinta siten, että se täyttää OWASP Application

Security Verification Standardin kappaleet V2 (Authentication) ja V3 (Session Management).

2.3. Cross-Site Scripting

Haavoittuvuuden merkitys

Cross-Site Scripting -haavoittuvuudet syntyvät, kun websovellus tulostaa käyttäjän antaman syötteen sellaisenaan websivuston rakenteeseen. Tällöin hyökkääjä kykenee sisällyttämään sivuston rakenteeseen esimerkiksi vihamielistä JavaScript-koodia. Vihamielinen koodi voidaan heijastaa websivustolle käyttäjän syötteestä (Reflected ja DOM based XSS) esimerkiksi URL:stä tai tallettaa pysyvästi palveluun (Stored XSS).

Haavoittuvuus johtaa sivustolla luottamuksellisuuden sekä eheyden menettämiseen. Hyökkääjä kykenee lukemaan kaiken tiedon, johon hyökkäyksen uhrilla on valtuudet tai muokata uhrille näytettävää sisältöä.

Haavoittuvuuden välttäminen

Haavoittuvuudelta voidaan välttyä suodattamalla kaikki syöte ennen verkkosivulle tulostusta. Useat viitekehykset tekevätkin tämän automaattisesti. Tutustu OWASP XSS Prevention Cheat Sheetiin, mikäli joudut syötettä suodattamaan manuaalisesti.

2.4. Turvattomat suorat objektiviittaukset

Haavoittuvuuden merkitys

Haavoittuvuus syntyy, kun sovellus käyttää suoraa viittausta ulkopuoliseen objektiin (esimerkiksi tietokantarivin id), eikä tarkista käyttäjän valtuutusta luettaessa tai muokattaessa kyseistä objektia. Muokkaamalla objektin arvoa (esimerkiksi muokkamalla tietokanta-id:n arvoa), hyökkääjä pääsee lukemaan tai muokkaamaan resursseja, joihin hänellä ei ole oikeuksia.

Haavoittuvuuden välttäminen

Varmenna sovelluksessasi, että jokainen pyyntö valtuutetaan sovelluksessa. Sovelluksen tulee varmentaa luettaessa tai muokattaessa resurssia käyttäjä valtuudet. Huomioithan, että tämä ei koske pelkästään tietokantatunnisteita, vaan myös esimerkiksi tiedostoja tai muita objekteja, joihin sovelluksessa viitataan käyttäen käyttäjän syötettä.

2.5. Turvattomat konfiguraatiot

Haavoittuvuuden merkitys

Haavoittuvuus syntyy, kun sovelluksen käyttämät alustaratkaisut sekä muut sovellukset kuten tietokannat, ovat turvattomasti konfiguroitu tai niitä ei ole päivitetty. Esimerkkejä tilanteista, jotka voivat johtaa haavoittuvuuden syntyyn:

- Turhien ominaisuuksien päällä pitäminen
- Sovelluksien sekä muiden komponenttien puutteellinen kovettaminen
- Päivittämättömät sovellukset
- Oletussalasanojen käyttö
- Virheviestien (stack traces) näyttäminen verkkosivustolla

Haavoittuvuuden välttäminen

Paras tapa välttää haavoittuvuuden syntyminen on implementoida päivitystenhallinta- sekä konfiguraationhallintaprosessit, joiden avulla kyetään pitämään sovellukset ajan tasalla sekä niiden konfiguraatiot turvallisina.

2.6. Arkaluonteisen tiedon paljastuminen

Haavoittuvuuden merkitys

Haavoittuvuus syntyy, kun arkaluonteista tietoa ei suojata säilytettäessä tai siirrettäessä riittävän hyvin. Puutteellinen tiedon suojaus voi johtaa tarpeettoman suureen määrään arkaluonteisten tietojen vuotamiseen tietomurron yhteydessä. Mikäli tunnistat jonkin alla olevista esimerkeistä, saatat olla haavoittuva tälle haavoittuvuudelle:

- Liian lyhyiden avainten pituuksien käyttö
- Salasanojen tallentaminen muussa muodossa kuin tiiviste-summana (suositeltavaa käyttää esimerkiksi PBKDF2-funktiota tai bcrypt-algoritmia)
- Heikkojen salausalgoritmien käyttäminen
- Tietoliikenteen suojaamattomuus arkaluonteista tietoa siirrettäessä

Haavoittuvuuden välttäminen

Paras tapa välttyä haavoittuvuudelta on analysoida palvelussa käsiteltävä tieto ja päättää tämän perusteella tarpeellinen suojaustaso. Tämän jälkeen sovelluskehityksen aikana huomioidaan, että arkaluonteinen tieto suojataan vaatimusten mukaisesti.

2.7. Puuttuva funktiotason pääsynhallinta

Haavoittuvuuden merkitys

Haavoittuvuus syntyy, kun palvelu ei varmenna käyttäjän valtuuksia päästä rajattuihin toiminnallisuuksiin. Tällöin hyökkääjä saattaa päästä näihin resursseihin käsiksi menemällä oikeaan URL:ään. Pahimmillaan hyväksikäyttö johtaa arkaluonteisen tiedon vuotamiseen tai eheyden

menettämiseen. Pahimmissa tapauksissa haavoittuvuus saattaa johtaa koko palvelun haltuunottoon, jos esimerkiksi hyökkääjä kykenee luomaan haavoittuvuuden avulla uusia käyttäjiä.

Haavoittuvuuden välttäminen

Sovellusta kehitettäessä tulee varmentaa käyttäjän valtuudet kaikkiin toiminnallisuuksiin ja resursseihin, joihin on rajattu pääsy. Sovelluksen auktorisointilogiikka tulee rakentaa siten, että oletuksena mihinkään resurssiin ei ole pääsyä, ellei käyttäjää ole luvitettu erikseen.

2.8. Pyyntöväärennös

Haavoittuvuuden merkitys

Haavoittuvuus syntyy, kun palvelu ei tarkista käyttäjän tekemän pyynnön alkuperää. Kun pyynnön alkuperää ei varmenneta, hyökkääjä voi suorittaa toimenpiteitä haavoittuvassa palvelussa houkuttelemalla uhrin vihamieliselle kolmannen osapuolen sivustolle.

Haavoittuvuus johtaa eheyden menettämiseen, sillä hyökkääjä voi suorittaa palvelussa toimenpiteitä ilman uhrin suostumusta. Pahimmassa tapauksessa lopputuloksena voi olla hyvin laajamittainen tietomurto, jos haavoittuvuuden avulla hyökkääjä voi luoda järjestelmään uusia käyttäjiä.

Haavoittuvuuden välttäminen

Haavoittuvuus on suositeltavaa korjata lisäämällä jokaiseen pyyntöön, joka muuttaa sovelluksen tilaa, uniikki tunniste. Tämä tunniste voidaan esimerkiksi luoda jokaisen istunnon alussa ja lisätä istunnon tietoihin. Kun pyyntö otetaan vastaan, vertaillaan pyynnön mukana tullutta tunnistetta istunnon tiedoissa olevaan tunnisteeseen. Mikäli tunnisteet eivät täsmää, on pyyntö väärennetty ja tulee se hylätä.

2.9. Haavoittuvien komponenttien käyttö

Haavoittuvuuden merkitys

Haavoittuvuus syntyy, kun sovelluksen käyttämiä kirjastoja sekä muita komponentteja ei pidetä ajantasalla. Haavoittuvuuden vaikutukset riippuvat täysin siitä, mitä komponentteja palvelussa on käytössä ja mitä haavoittuvuuksia niissä on. Pahimmassa tapauksessa hyväksikäyttö voi johtaa palvelun täydelliseen murtamiseen. Yleisimmissä tapauksissa kyse on hieman rajatummista ongelmista, kuten Cross-Site Scripting -haavoittuvuuksista.

Haavoittuvuuden välttäminen

Jotta haavoittuvuuden syntymiseltä voidaan välttyä, tulee kaikki sovelluksen käyttämät komponentit ja kirjastot sekä niiden versionumerot dokumentoida. Tämän jälkeen tulee päivityshallintaprosessiin ottaa mukaan kirjastojen ja komponenttien päivitystilanteen seuranta ja tarvittaessa päivittää niitä, mikäli käytössä olevista komponenteista löytyy haavoittuvuuksia.

2.10. Validoimattomat edelleenohjaukset

Haavoittuvuuden merkitys

Haavoittuvuus syntyy, kun sovellus käyttää käyttäjän syötettä edelleenohjauksen kohteen määrittelyyn. Tällöin hyökkääjä pystyy antamaan haavoittuvaan kohdepalveluun osoittavan linkin, joka kuitenkin ohjaa uhrin selaimen kolmannen osapuolen sivustolle. Haavoittuvuuden avulla hyökkääjä kykenee esimerkiksi suorittamaan kalasteluhyökkäyksiä – vihamieliselle verkkosivulle voi esimerkiksi toteuttaa kirjautumisikkunan, joka on identtinen haavoittuvan kohdepalvelun kanssa.

Haavoittuvuuden välttäminen

Käyttäjän syöte tulee varmentaa ennen selaimen uudelleenohjaamista ja tarkistaa, että uudelleenohjauksen kohde on sallittu ja luotettava palvelu. Yleensä tämä voidaan toteuttaa ns. whitelist-menetelmällä, jossa on listattu sallitut uudelleenohjauksen kohteet.

3. Yhteenveto

Esitellyt haavoittuvuudet ovat kokemuksemme ja OWASP Top 10 -listan mukaan kriittisimmät websovelluksista löytyvät haavoittuvuudet. Kehittämällä sovelluksen siten, että edellä mainitut haavoittuvuudet ovat huomioitu ja kontrollit implementoitu, saadaan verkkosovelluksista huomattavasti turvallisempia. Verkkosovelluksia kehitettäessä muista seuraavat asiat:

1. Validoi käyttäjän syöte

2. Auktorisoi käyttäjän tekemät pyynnöt

3. Analysoi sovelluksen käsittelemä tiedon luonne ja suunnittele kontrollit tieto-omaisuuden suojaamiseksi

4. Tunnista käyttäjät käyttäen riittävän vahvaa tunnistamistapaa huomioiden sovelluksen luonteen

2NS tarjoaa tukea sovelluskehityksen elinkaaren eri vaiheissa sekä tietoturvan varmentamisessa. Autamme sinua mielellämme sovelluksien tietoturvan parantamisessa.

www.2ns.fi

Second Nature **Security**

2NS on vuodesta 2005 asti toteuttanut yli 1000 tietoturva-auditointia ja haavoittuvuustestausta.

Palvelemme yli 150 koti- ja ulkomaista asiakasta (Suomi, USA, Ruotsi, Hollanti, Viro, Latvia, Norja). Asiakaskuntamme koostuu muun muassa yrityksistä, julkishallinnosta ja finanssialasta.

Palvelemme asiakkaitamme tinkimättömällä ammattitaidolla. Meille on tärkeää, että asiakas saa tulokset ymmärrettävässä muodossa selkeiden korjauskehoitusten kera.

Meillä on kokemusta valtakunnan kriittisimmistä ympäristöistä: Asiakkaamme ovat pienistä ohjelmistoyrityksistä suurempiin toimijoihin, muun muassa Finnair, Neste Oil, Veikkaus, CSC, VRK, Affecto, Innofactor, Tallink Silja, Huhtamäki ja Varma.

Haavoittuvuustutkimustiimimme on julkaissut lukuisia haavoittuvuustiedotteita, muun muassa SAP:n, Oraclen, F-Securen, IBM:n ja HP:n laitteista ja järjestelmistä.

Ota yhteyttä, autamme mielellämme!

www.2ns.fi

Second Nature Security Oy www.2ns.fi | 010 322 9000 | info@2ns.fi