

Input Validation 1

Zoals altijd kunnen de oefeningen binnengehaald worden door een git pull te doen.

```
git pull
```

of de git repository te clonen als je deze nog niet hebt.

```
git clone https://github.com/similonap/software_security_2021.git
```

Wat ga je leren in dit labo?

- Gebruik maken van Chrome Developer Tools
- Het nut van server side input validation/output sanitization begrijpen
- Code injection / SQL Injection
- Regular expressions

Stappenplan

1. Ga naar de `labo_validation` directory en doe vervolgens

```
npm install
```

2. In deze directory staat een heel onveilige ticket reservatie applicatie. Je kan deze opstarten met

```
node index.js
```

en dan naar `http://localhost:3000` te surfen.

3. Je komt dan op een login pagina. Je kan eens proberen in te loggen met

```
username: joske  
password: hunter2
```

vergeet niet de captcha in te vullen door de puzzel op te lossen. (een simpele som)

4. Je kan hier een ticket kopen voor een fictief festival. Hier zijn een aantal regels vastgelegd die worden gevalideerd op je browser.

Kijk naar de bron code in chrome developer tools en geef hier een overzicht van welke regels er gelden voor het ticketten formulier:

Naam: verplicht maar read-only,
enkel a-z of A-Z toegelaten
minimaal 1 karakter en geen maximum

Aantal tickets: verplicht in te vullen met een waarde van
1 tem 3

Land: verplicht in te vullen 2 opties:

Tip: Er wordt gebruik gemaakt van regular expressions. Kan je die niet lezen kan je ze gewoon uitproberen op <https://regex101.com/>

Wees volledig!

5. Gebruik de chrome developer tools om nu de validatie regels van het formulier aan te passen zodat je

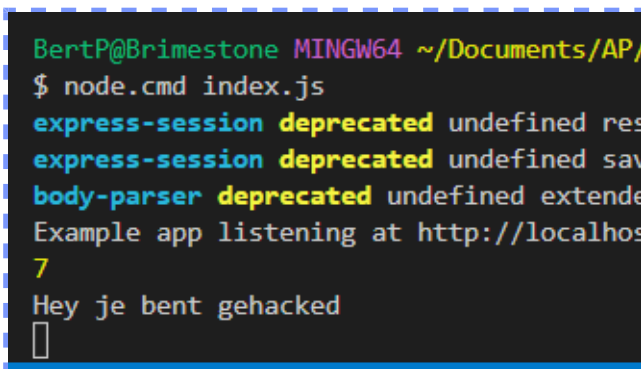
- Tickets kan bestellen op je eigen naam, zelfs al ben je ingelogd met Joske
- 10 Tickets kan bestellen in plaats van het maximum aantal tickets.
- Tickets bestellen met landscode FR (Frankrijk)

Neem een screenshot waar duidelijk staat dat je de tickets hebt gekocht en de chrome developer tools op te zien zijn. Sleep deze hieronder in:



6. We gaan nu een code injection aanval proberen te doen zodat we wat te weten kunnen komen over de server. Log uit en pas de broncode zodat je het veld `captchaSource` toch kan aanpassen (want die is nu readonly).

Zorg ervoor dat er in het log venster van de server 'Hey je bent gehacked' komt te staan. Neem hiervan een screenshot en sleep deze hieronder in"



Zoek in de `index.js` bron code waarom je zomaar code kan uitvoeren via dit veld. Leg hieronder uit waarom dit is.

Ik heb geen code uitgevoerd
Ik de console print altijd de ontvangen input uit
dus als de browser niet controleert (set number -> text)
dan kan je eender wat "uitprinten" in de server console.

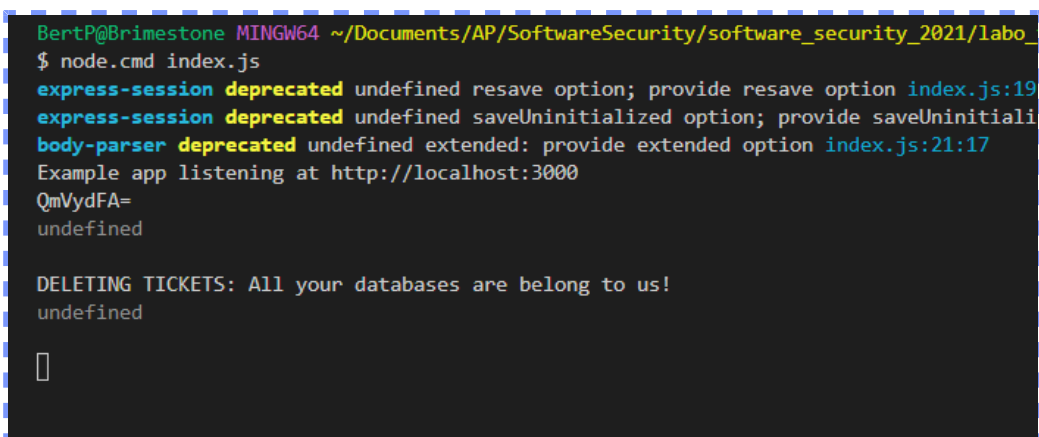
7. Zorg dat de variabele `SESSION_SECRET` aan ons getoond wordt aan de hand van het onveilige captcha veld.
Wat is inhoud van deze variabele:

QmVydFA=

Dit zal er ongeveer uitzien als `QW5kaWUgU2ltaWxvbg==`

8. De code van `index.js` bevat een functie die de tabel van de ticketten leegmaakt. Probeer nu ook deze functie aan te roepen via dit onveilige captcha veld.

Er zal iets in je log van je terminal komen. Neem hier een screenshot van en sleep dit hieronder in:



```
BertP@Brimestone MINGW64 ~/Documents/AP/SoftwareSecurity/software_security_2021/labo_
$ node.cmd index.js
express-session deprecated undefined resave option; provide resave option index.js:19
express-session deprecated undefined saveUninitialized option; provide saveUninitiali
body-parser deprecated undefined extended: provide extended option index.js:21:17
Example app listening at http://localhost:3000
QmVydFA=
undefined

DELETEING TICKETS: All your databases are belong to us!
undefined

█
```

9. Probeer nu aan de hand van de `require()` functie in javascript de code van 'magicword.js' te laten uitvoeren op de server. De server zal mogelijk vastlopen (CTRL-C om terug te beginnen).

Er zal iets in je log van je terminal komen. Neem hier een screenshot van en sleep dit hieronder in:

```
at next (C:\Users\BertP\Documents\AP\Software
at Route.dispatch (C:\Users\BertP\Documents\A
at Layer.handle [as handle_request] (C:\Users
5)
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
{}

AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
AH AH AH! YOU DIDN'T SAY THE MAGIC WORD!
```

10. Pas de code aan van `index.js` zodat er geen ticketten meer kunnen besteld worden met een foute naam, verkeerde hoeveelheid en foute landcodes.

Tip: Je hebt hier enkel een paar if statements voor nodig. Zoek naar de TODO van server validation.

11. Pas de code aan van `index.js` zodat de captcha niet meer kan misbruikt worden voor code injection. Het mag alleen maar sommen bevatten en geen andere code.

Tip: Je moet input validatie op `req.body.captchaSource` doen aan de hand van een regular expression. Je mag zelf kiezen welke error message je terug geeft.

12. Print deze pagina af als PDF en slaag deze op als `naam_voornaam_labov_validation.pdf` en stuur de volgende files op via digitap:

- `naam_voornaam_labov_validation.pdf`
- `tickets.db`
- `index.js`