client-controlled identifier & Path Traversal

A01:2021-Broken Access Control

Broken Access Control

Users can access things they should NOT be allowed to access.

Routes and files

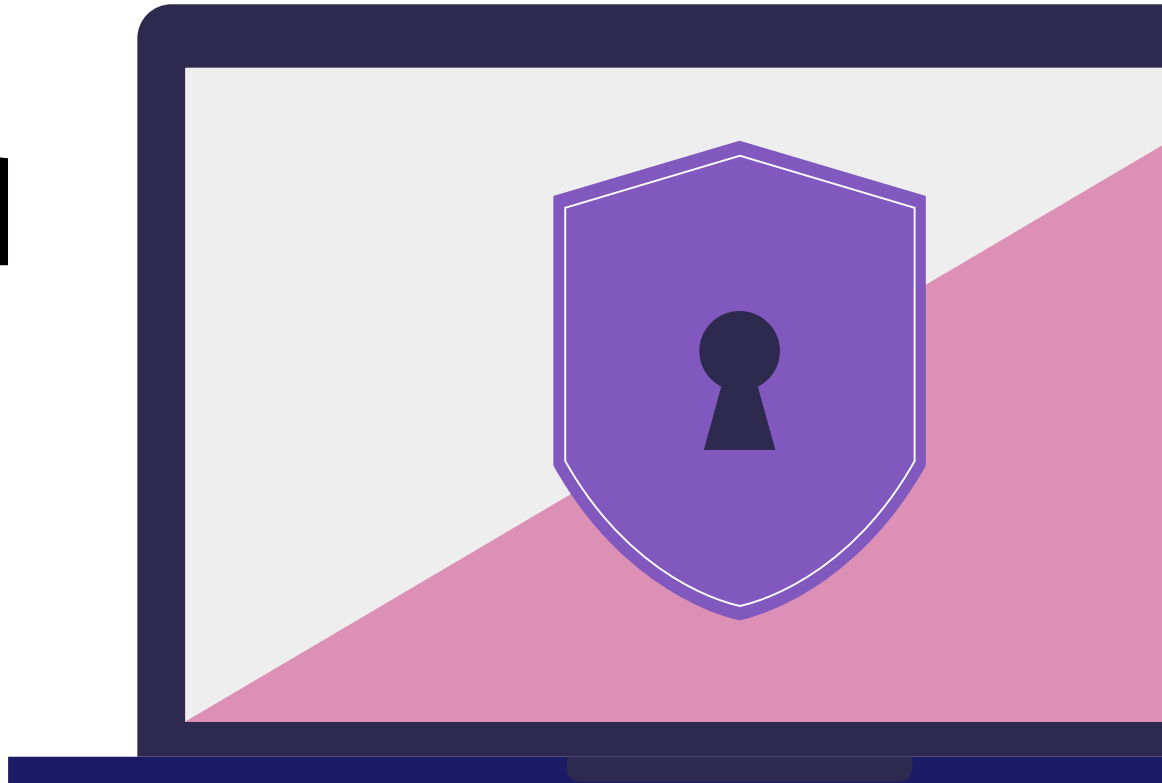
Files

- frontend.js
- Order.js

Routes

- /v1/beer-pic/
- http://localhost:5000/profile?id={number}

client-controlled identifier



vulnerable code

```
app.get('/profile', (req,res) =>{

    if(!req.query.id){
        res.redirect("/?message=Could not Access profile please log in or register")
        return;
    }
    const user = db.user.findAll({include: // Notice `include` takes an ARRAY
        'beers',
        where: {
            id: req.query.id
        }}).then(user => {
        if(user.length == 0){
            res.redirect('/?message=User not found, please log in')
            return;
        }
        let beers = db.beer.findAll().then(beers => {
        res.render('profile.html',{beers : beers, user:user[0]});
     })
});
});
```

exploitation

you can change in the url:

- Route: http://localhost:5000/profile?id=1
- Route: http://localhost:5000/profile?id=2
- Route: http://localhost:5000/profile?id=3

and still login even if you dont have credentials for those accounts

---

explanation

The application exposes user profile access through a client-controlled identifier (id) in the URL, without enforcing proper authorization checks on the server side.
This allows users to access profiles that do not belong to them simply by modifying the profile ID value in the request.

Semgrep rule

```yaml
rules:
  - id: idor-client-controlled-id
    patterns:
      - pattern: req.query.id
    message: "Client-controlled ID (Broken Access Control)"
    severity: ERROR
    languages: [javascript]

```

Fixes

```
    app.get('/profile', (req,res) =>{

        if(!req.query.id){
            res.redirect("/?message=Could not Access profile please log in or register")
            return;
        }
        const user = db.user.findAll({include: // Notice `include` takes an ARRAY
            'beers',
            where: {
                id: req.query.id
            }}).then(user => {
            if(user.length == 0){
                res.redirect('/?message=User not found, please log in')
                return;
            }
            let beers = db.beer.findAll().then(beers => {
            res.render('profile.html',{beers : beers, user:user[0]});
         })
    });
});
```

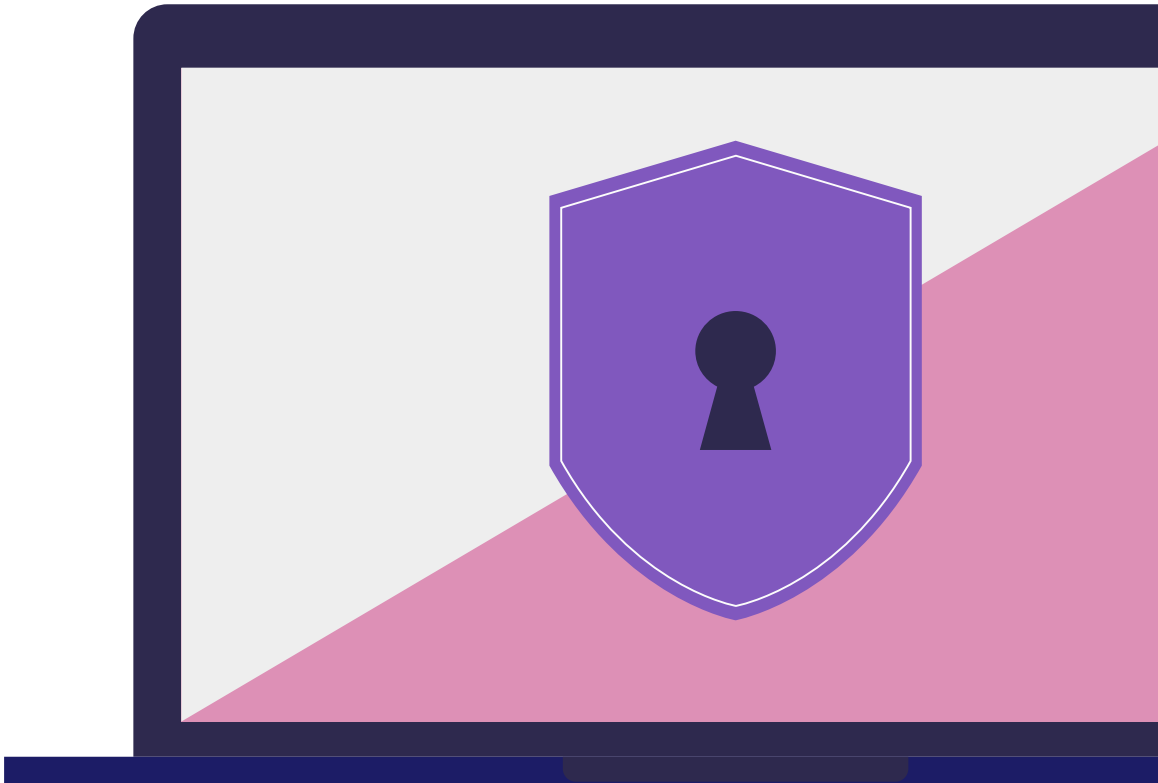
```
app.get('/profile', async (req, res) => {

    if (!req.session.logged || !req.session.userId) {
        return res.redirect('/?message=Please log in');
    }

    try {
        if (!req.session.logged || !req.session.userId) {
            return res.redirect('/?message=Please log in');
        }
        const user = await db.user.findOne({
            //where: { id: req.session.userId },
             where:{ id: req.session.userId },
            include: 'beers'
        });

        if (!user) {
            return res.redirect('/?message=User not found');
        }

        const beers = await db.beer.findAll();

        res.render('profile.html', {
            user,
            beers
        });

    } catch (err) {
        console.error(err);
        res.status(500).send("Internal Server Error");
    }
});
```

Fixes

```
if(user[0].password == md5(userPassword)){
        req.session.logged = true
        res.redirect('/profile?id='+user[0].id);
        return;
}
```

```
if(user[0].password == md5(userPassword)){
        req.session.logged = true;
        req.session.userId = user[0].id;
        res.redirect('/profile');
        return;
}
```

Path Traversal



vulnerable code

```
app.get('/v1/beer-pic/', (req,res) =>{
    var filename = req.query.picture,
    filePath = `../../../uploads/${filename}`;
    const path=require('path')
    //console.log(__dirname)
    //console.log(path.dirname(filePath))
    //path.normalize(filePath)
    fs.readFile(path.join(__dirname, filePath),function(err,data){
    if (err){
        res.send("error")
    }else{
        if(filename.split('.').length == 1){
           res.type('image/jpeg')
           //res.set('Content-Type', 'image/jpg');
           res.send(data)
           return;
         }
     let buffer = Buffer.from(data, 'utf8');
     res.send(buffer)
     }
   })
 });
```

exploitation

http://localhost:5000/v1/beer-pic/?picture=../../../package.json in postman

---

explanation

- The server's normal behavior is that the http://localhost:5000/v1/beer-pic/?picture={pic} returns the photo of a beer (ex: http://localhost:5000/v1/beer-pic/?picture=bud.jpg) so you can view/download it.

- The vulnerability here is that the server trusts users to provide safe filenames, but users can provide malicious paths like ../../../{filename} to read any file on the server! The server directly concatenates user input into a file path without validation: ../../../uploads/${filename}. When filename contains directory traversal sequences (../), it escapes the intended uploads directory.

- For example: http://localhost:5000/v1/beer-pic/?picture=../../../package.json becomes ../../../uploads/../../../package.json, which resolves to ../../package.json, allowing reading of the project's package.json file from the server's filesystem.

Semgrep rule

```yaml
rules:
  - id: any-fs-read-with-user-input
    message: File read with user input - Path Traversal risk
    severity: ERROR
    languages: [javascript]
    pattern: |
      fs.readFile($PATH, ...)
```

fixes

```
app.get('/v1/beer-pic/', (req,res) =>{
    var filename = req.query.picture,
    filePath = `../../../uploads/${filename}`;
    const path=require('path')
    //console.log(__dirname)
    //console.log(path.dirname(filePath))
    //path.normalize(filePath)
    fs.readFile(path.join(__dirname, filePath),function(err,data){
    if (err){
        res.send("error")
    }else{
        if(filename.split('.').length == 1){
          res.type('image/jpeg')
          //res.set('Content-Type', 'image/jpg');
          res.send(data)
          return;
        }
     let buffer = Buffer.from(data, 'utf8');
     res.send(buffer)
     }
   })
 });
```

```
app.get('/v1/beer-pic/', (req,res) =>{
    const filename = req.query.picture;

    // Validate input - reject path traversal
    if (filename.includes('..') || filename.includes('/') || filename.includes('\\')) {
        return res.status(400).send("Invalid filename");
    }

    // Allow only image files
    const allowedExtensions = ['.jpg', '.jpeg', '.png'];
    const ext = require('path').extname(filename).toLowerCase();
    if (!allowedExtensions.includes(ext)) {
        return res.status(400).send("Invalid file type");
    }

    // Read and send file
    const filePath = require('path').join(__dirname, '../../../uploads', filename);
    require('fs').readFile(filePath, function(err, data){
        if (err) return res.status(404).send("Not found");
        res.type(ext === '.png' ? 'image/png' : 'image/jpeg').send(data);
    });
});
```