NODE.JS GLOBAL

Authentication. Security. CORS

FEBRUARY, 2020

AGENDA

1. Authentication with Passport.js
2. JSON Web Tokens (JWT)
3. Security
4. CORS

AUTHENTICATION

1. Passport.js
2. JSON Web Tokens

WHAT IS PASSPORT

1. Authentication middleware for Node.js
2. Extremely flexible and modular
3. Can be unobtrusively used with any Express application
4. 500+ strategies to choose from

GET IT WORK

1. Require passport module and initialize it
2. Configure passport with at least one Strategy
3. Specifying a route for authentication
4. Protect routes

SETUP LOCAL STRATEGY

```
// setup local strategy
passport.use(new LocalStrategy({
    usernameField: 'firstName',
    passwordField: 'lastName',
    session: false
}, function (username, password, done) {
    let employee = _.find(data, { firstName: username });

    if (employee === undefined || employee.lastName !== password) {
        done(null, false, 'Bad username/password combination');
    } else {
        done(null, employee);
    }
  }
));
// setup bearer strategy
passport.use(new BearerStrategy(
    function (token, done) {
        let result = _.find(tokens, { token: token });

        if (result === undefined) {
            done(null, false);
        } else {
            done(null, result, { scope: 'all' })
        }
    }
));
```

```
// a piece of JSON with data
{
    "isActive": true,
    "title": "Senior Software Engineer",
    "lastName": "Williamson",
    "firstName": "Marsh",
    "id": "cdb24955-4de2-4ae2-9419-2dd509027de6"
},
```

```
// a piece of JSON with tokens
{
    "id": "cdb24955-4de2-4ae2-9419-2dd509027de6",
    "token": "TOKEN-6e87-4baf-b942-f984d9cd0635"
},
```

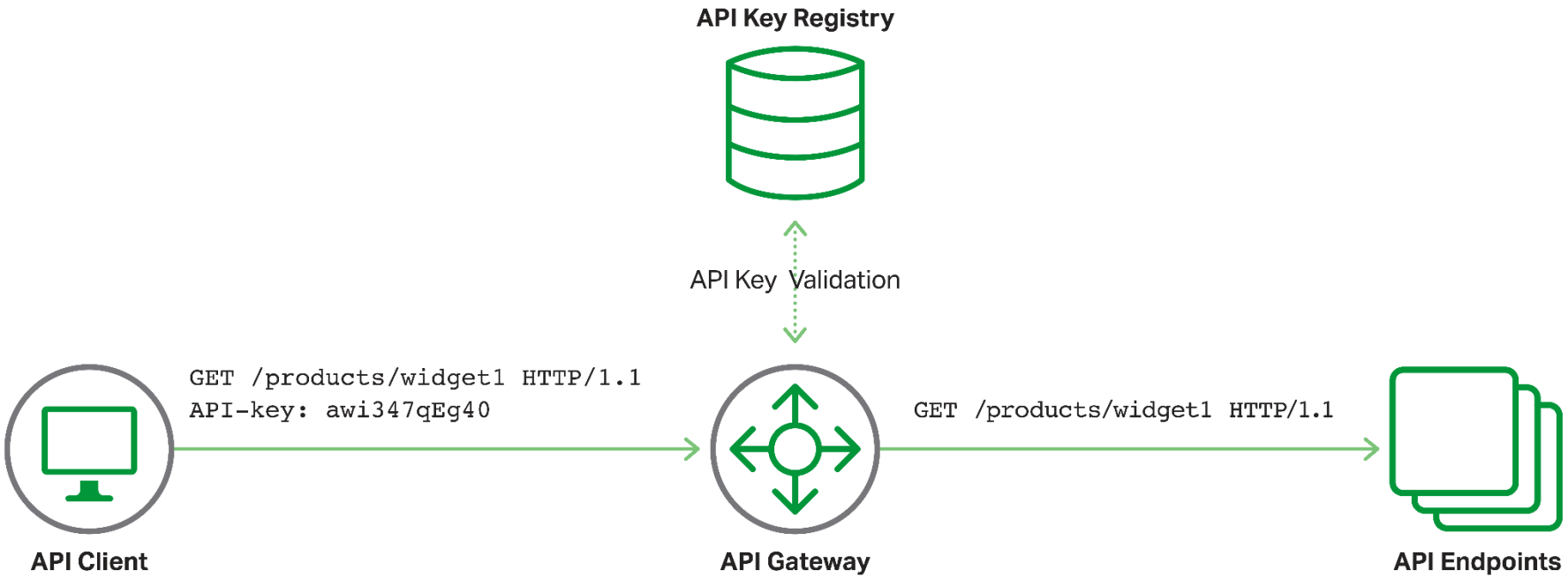
CONNECT WITH EXPRESS

```
// setup authentication route with local strategy
app.post('/authenticate', passport.authenticate('local', { session: false }), function (req, res) {
        let token = _.find(tokens, { id: req.user.id });

        res.json(token);
    }
);

// protect endpoint with bearer strategy
app.get('/employees', passport.authenticate('bearer', { session: false }), function (req, res) {
    res.json(data);
});
```

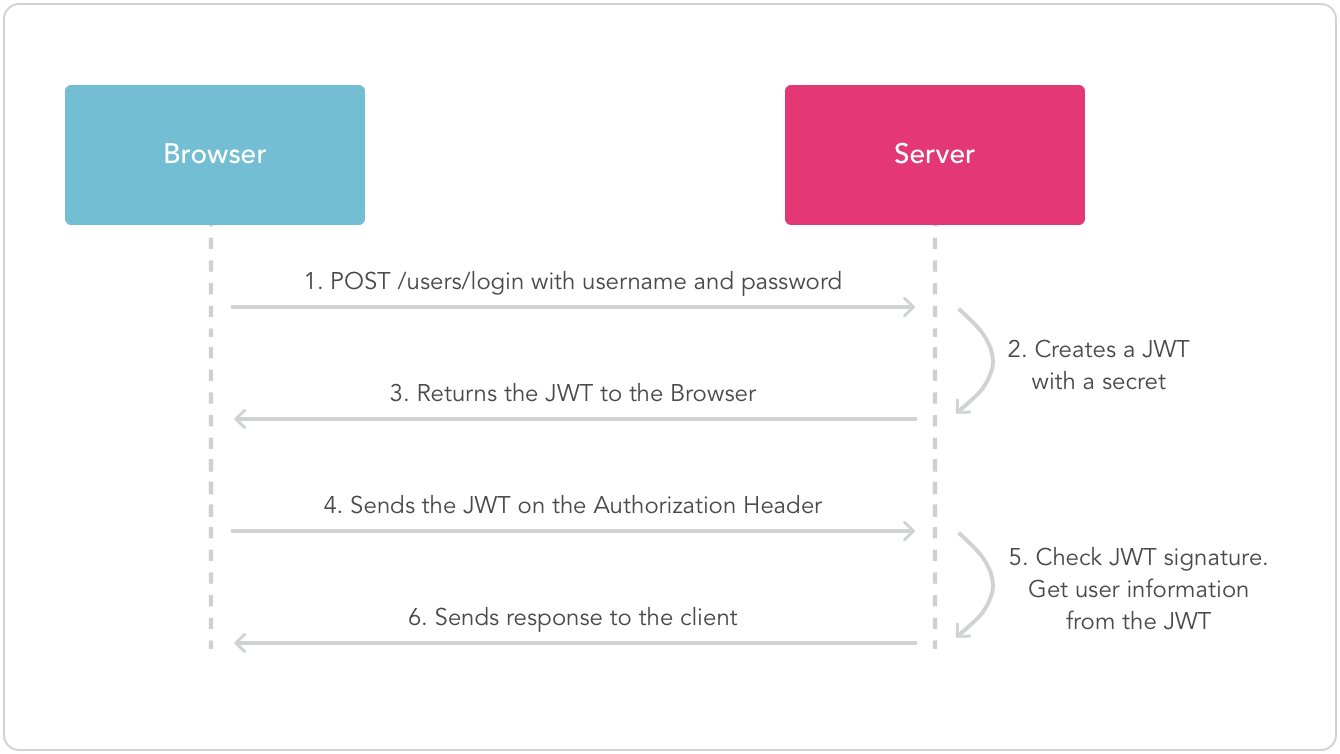
REQUESTS

1. POST request to the authentication route with data in body
   As a result get JSON with ID and TOKEN in response
2. GET request to the protected authentication with bearer token

REGULAR TOKEN AUTH



HOW DOES IT WORK



COMPACT AND SELF-CONTAINED

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) secret base64 encoded
```

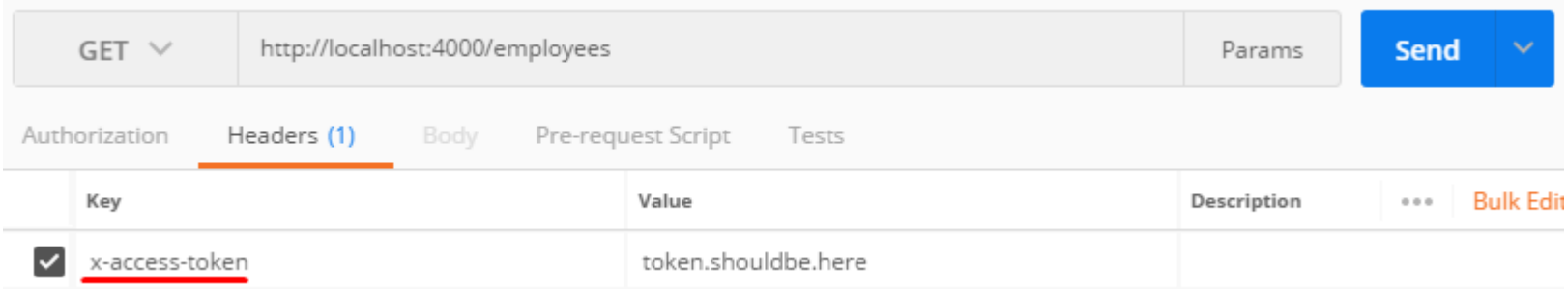
* https://jwt.io/

SETUP JWT

```
app.post('/authenticate ', function (req, res) {
    let employee = _.find(data, { firstName: req.body.firstName });

    if (employee === undefined || employee.lastName !== req.body.lastName) {
        res.status(403).send({ success: false,  message: 'Bad username/password combination.' });
    } else {
        let payload = { "sub": employee.id, "isActive": employee.isActive };
        let token = jwt.sign(payload, 'secret', { expiresIn: 10 });
        res.send(token);
    }
});

// middleware for token check
function checkToken(req, res, next) {
    let token = req.headers['x-access-token'];

    if (token) {
        jwt.verify(token, 'secret', function(err, decoded) {
            if (err) {
                res.json({ success: false, message: 'Failed to authenticate token.' });
            } else {
                // some business logic here
                next();
            }
        });
    } else {
        res.status(403).send({ success: false, message: 'No token provided.' });
    }
}
```

PROTECT ENDPOINT WITH JWT

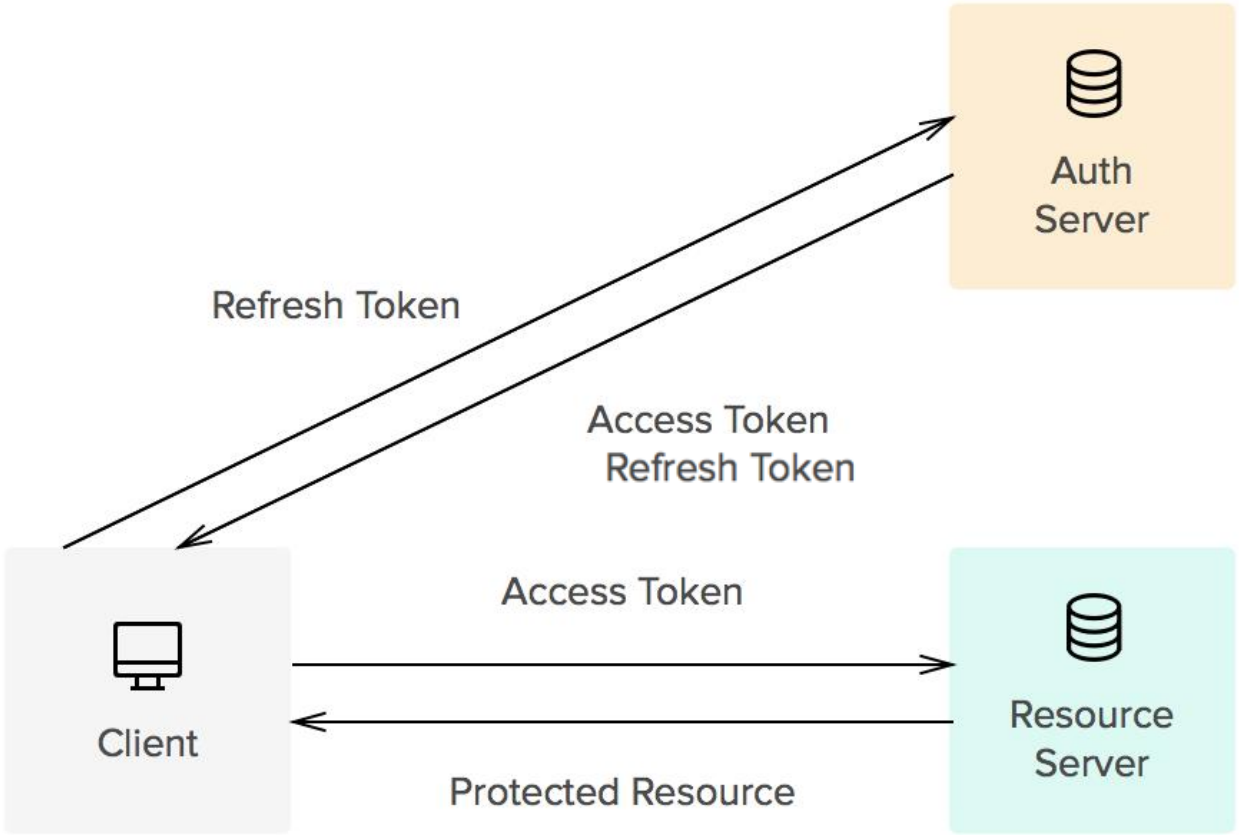
```
// middleware for token check
function checkToken(req, res, next) {
    let token = req.headers['x-access-token'];

    if (token) {
        jwt.verify(token, 'secret', function(err, decoded) {
            if (err) {···
            } else {
                // some business logic here
                next();
            }
        });
    } else {···
    }
}

// protect endpoint with JWT token
app.get('/employees', checkToken, function (req, res) {
    res.json(data);
});
```



JWT PROS AND CONS

- No server-side storage
- Easy to use (a lot of libs)
- Can be used across services
- One key rules all
- Can't be revoked permanently (in general)

ACCESS TOKEN AND REFRESH TOKEN



OWASP TOP 10

OWASP Top 10 2013	OWASP Top 10 2017
A1 Injection	A1 Injection
A2 Broken Auth and Session Management	A2 Broken Authentication
A3 Cross-Site Scripting (XSS)	A3 Sensitive Data Exposure
A4 Insecure Direct Object References	A4 XML External Entities (XXE)
A5 Security Misconfiguration	A5 Broken Access Control
A6 Sensitive Data Exposure	A6 Security Misconfiguration
A7 Missing Function Level Access Control	A7 Cross-Site Scripting (XSS)
A8 Cross-Site Request Forgery (CSRF)	A8 Insecure Deserialization
A9 Using Components with Known Vulnerabilities	A9 Using Components with Known Vulnerabilities
A10 Unvalidated Redirects and Forwards	A10 Insufficient Logging & Monitoring

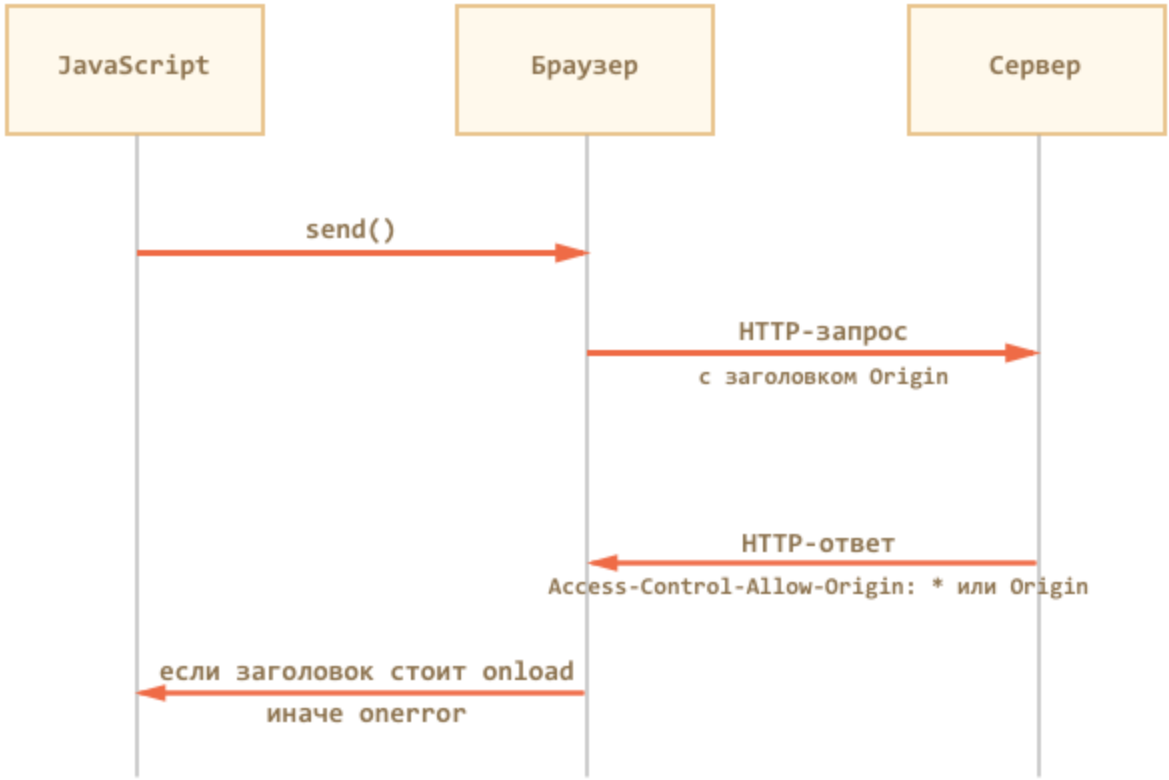
NODE SECURITY

1. Static analysis
2. Sensitive data on the client side
3. Running process with superuser rights
4. Vulnerabilities in dependencies
5. Known vulnerabilities in your code
6. Error logging

API SECURITY

1. HTTPS
2. Access Control
3. Restrict HTTP methods
4. Input validation
5. Error handling
6. Audit logs
7. Security headers
8. Sensitive information in HTTP requests

CORS


JavaScript
Браузер
Сервер
send()
HTTP-запрос
с заголовком Origin
HTTP-ответ
Access-Control-Allow-Origin: * или Origin
если заголовок стоит onload
иначе onerror

HELMET

1. Hide Powered-By
2. HSTS
3. No Cache
4. .. and more

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

app.use(helmet())
```

HELMET

USEFUL LINKS

- Custom Passport strategy
- node-jsonwebtoken
- Open Web Application Security Project (OWASP)
- REST API Security Cheat Sheet
- Express Production Security Best Practices
- Retire.js, Node Security Platform, Snyk
- Helmet
- CORS, CORS lib

NODE.JS GLOBAL

AUTHENTICATION. SECURITY. CORS

BY
VLADISLAV LOMAKO
HERMAN BILOUS