:iTechArt

Software
Engineering
Services

PASSPORT JS

Anton Chernov

WHAT IS PASSPORT JS?

→ Middleware для авторизации в Node JS

ОСОБЕННОСТИ **PASSPORT JS**

- 300+ authentication strategies
- Single sign-on with OpenID and OAuth
- Easily handle success and failure
- Supports persistent sessions
- Dynamic scope and permissions
- Pick and choose required strategies
- Implement custom strategies
- Does not mount routes in application
- Lightweight code base

INSTALLATION AND USAGE

→ npm install passport or yarn add passport

CONFIGURE

1. Authentication strategies
2. Application middleware
3. Sessions (*optional*)

MIDDLEWARE

CODE SNIPPET

```
app.configure(function() {
  app.use(express.static('public'));
  app.use(express.cookieParser());
  app.use(express.bodyParser());
  app.use(express.session({
    secret: 'keyboard cat'
  }));
  app.use(passport.initialize());
  app.use(passport.session());
  app.use(app.router);
});
```

COOKIES

CODE SNIPPET

```
Set-Cookie: <name>=<value>[;
<Max-Age>=<age>]
[; expires=<date>]
[; domain=<domain_name>]
[; path=<some_path>]
[; secure][; HttpOnly]
```

SESSIONS

CODE SNIPPET

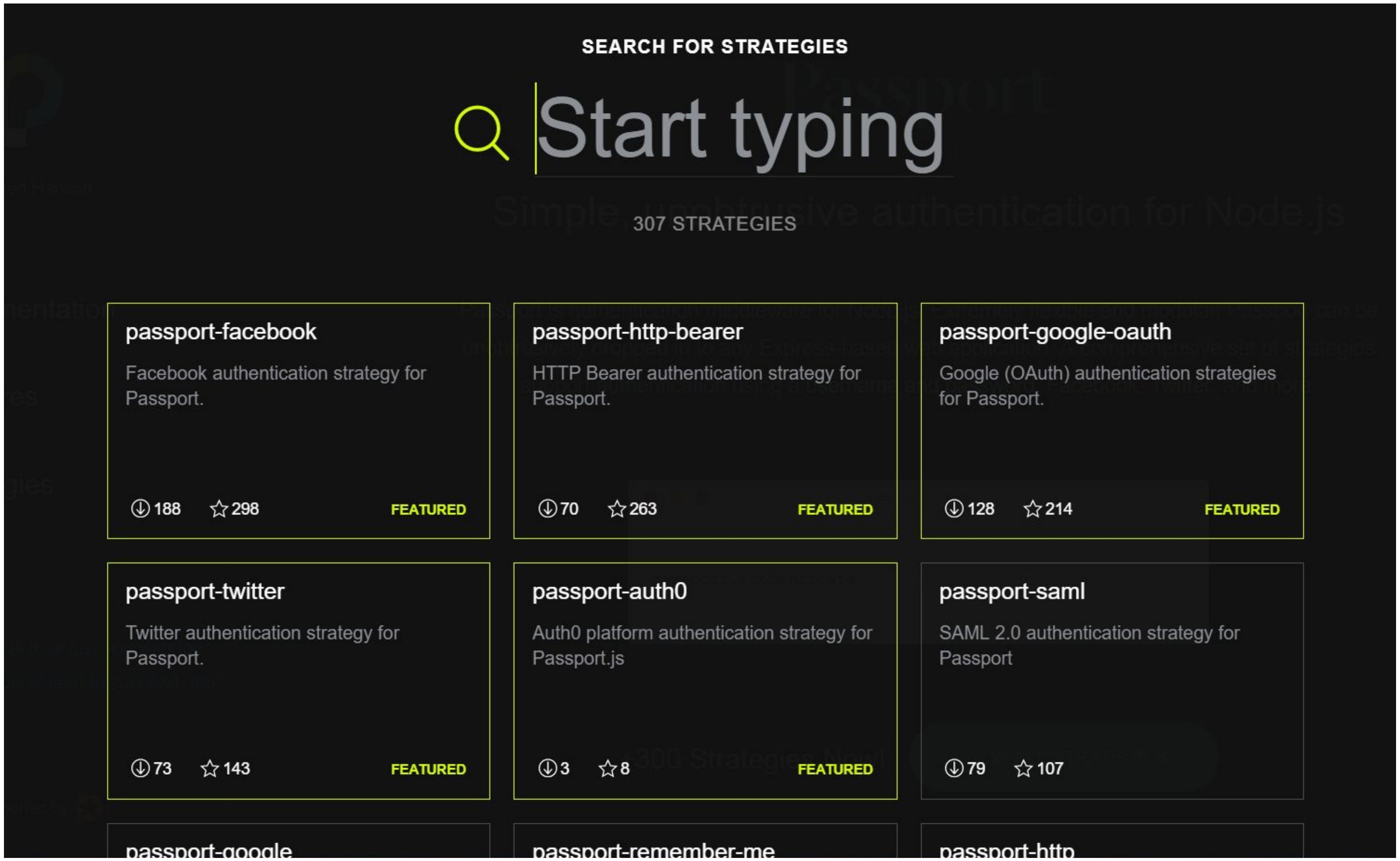
```
passport.serializeUser(function(user, done) {
  done(null, user.id);
});

passport.deserializeUser(function(id, done) {
  User.findById(id, function(err, user) {
    done(err, user);
  });
});
```

DISABLE SESSIONS

CODE SNIPPET

```
app.get('/api/users/me', passport.authenticate('basic', {
  session: false
}), function (req, res) {
  res.json({
    id: req.user.id,
    username: req.user.username
  });
});
```

СТРАТЕГИИ АУТЕНТИФИКАЦИИ



SEARCH FOR STRATEGIES

Start typing

307 STRATEGIES

passport-facebook
Facebook authentication strategy for Passport.
188 ☆298 FEATURED

passport-http-bearer
HTTP Bearer authentication strategy for Passport.
70 ☆263 FEATURED

passport-google-oauth
Google (OAuth) authentication strategies for Passport.
128 ☆214 FEATURED

passport-twitter
Twitter authentication strategy for Passport.
73 ☆143 FEATURED

passport-auth0
Auth0 platform authentication strategy for Passport.js
3 ☆8 FEATURED

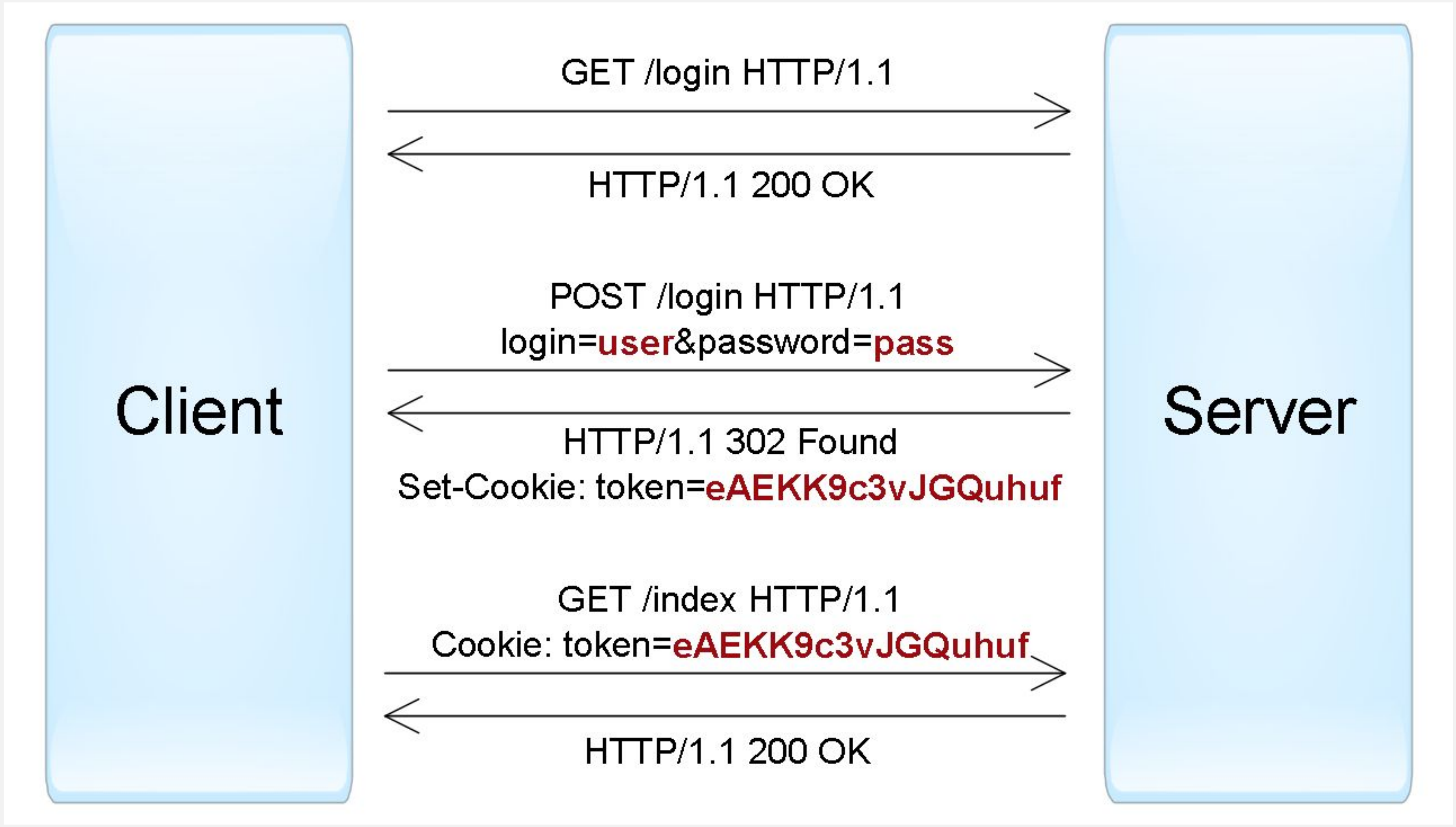
passport-saml
SAML 2.0 authentication strategy for Passport
79 ☆107

passport-google

passport-remember-me

passport-http

FORM BASED AUTHENTICATION



LOCAL STRATEGY

CODE SNIPPET

```
var passport = require('passport'), LocalStrategy =
require('passport-local').Strategy;
passport.use(new LocalStrategy(function (username, password, done) {
  User.findOne({username: username}, function (err, user) {
    if (err) {
      return done(err);
    }
    if (!user) {
      return done(null, false,
        {message: 'Incorrect username.'});
    }
    if (!user.validPassword(password)) {
      return done(null, false,
        {message: 'Incorrect   password.'});
    }
    return done(null, user);
  });
}));
```

LOCAL STRATEGY CONFIGURATION

CODE SNIPPET

```
passport.use(new LocalStrategy({
  usernameField: 'email',
  passwordField: 'passwd'
}, function (username, password, done) {
  // ...
}));
```

LOCAL STRATEGY FORM

CODE SNIPPET

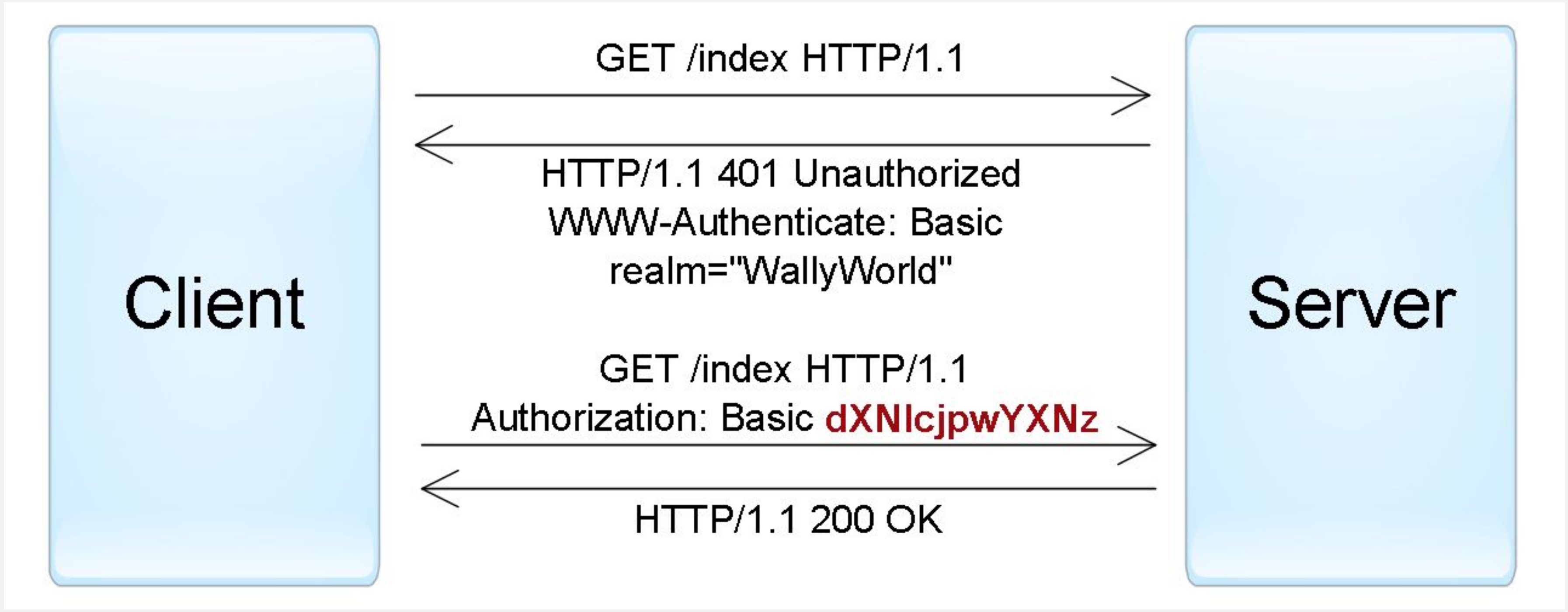
```
<form action="/login" method="post">
    <div> <label>Username:</label> <input type="text" name="username" /> </div>
    <div> <label>Password:</label> <input type="password" name="password" /> </div>
    <div> <input type="submit" value="Log In" /> </div>
</form>
```

AUTHENTICATING REQUESTS

CODE SNIPPET

```
app.post('/login',
  passport.authenticate('local'),
  function (req, res) {
    // If this function gets called, authentication was
    // successful.
    // `req.user` contains the authenticated user.
    res.redirect('/users/' + req.user.username);
  });
```

HTTP BASIC AUTHENTICATION

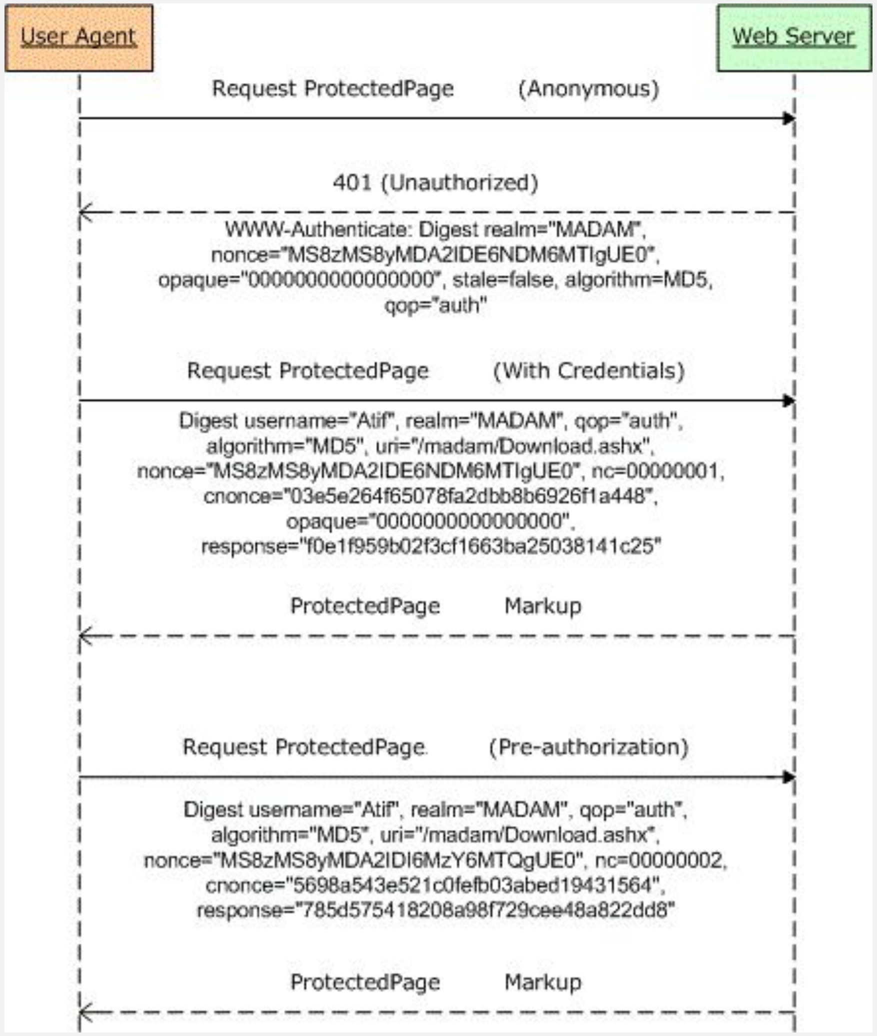


BASIC STRATEGY

CODE SNIPPET

```
passport.use(new BasicStrategy(
  function(username, password, done) {
    User.findOne({ username: username }, function (err, user) {
      if (err) { return done(err); }
      if (!user) { return done(null, false); }
      if (!user.validPassword(password)) { return done(null, false); }
      return done(null, user);
    });
  }
));
```

HTTP DIGIST AUTHENTICATION


User Agent
Web Server
Request ProtectedPage (Anonymous)
401 (Unauthorized)
Request ProtectedPage (With Credentials)
ProtectedPage Markup
Request ProtectedPage (Pre-authorization)
ProtectedPage Markup

```
WWW-Authenticate: Digest realm="MADAM",
nonce="MS8zMS8yMDA2IDE6NDM6MTIgUE0",
opaque="0000000000000000", stale=false, algorithm=MD5,
qop="auth"
```

```
Digest username="Atif", realm="MADAM", qop="auth",
algorithm="MD5", uri="/madam/Download.ashx",
nonce="MS8zMS8yMDA2IDE6NDM6MTIgUE0", nc=00000001,
cnonce="03e5e264f65078fa2dbb8b6926f1a448",
opaque="0000000000000000",
response="f0e1f959b02f3cf1663ba25038141c25"
```

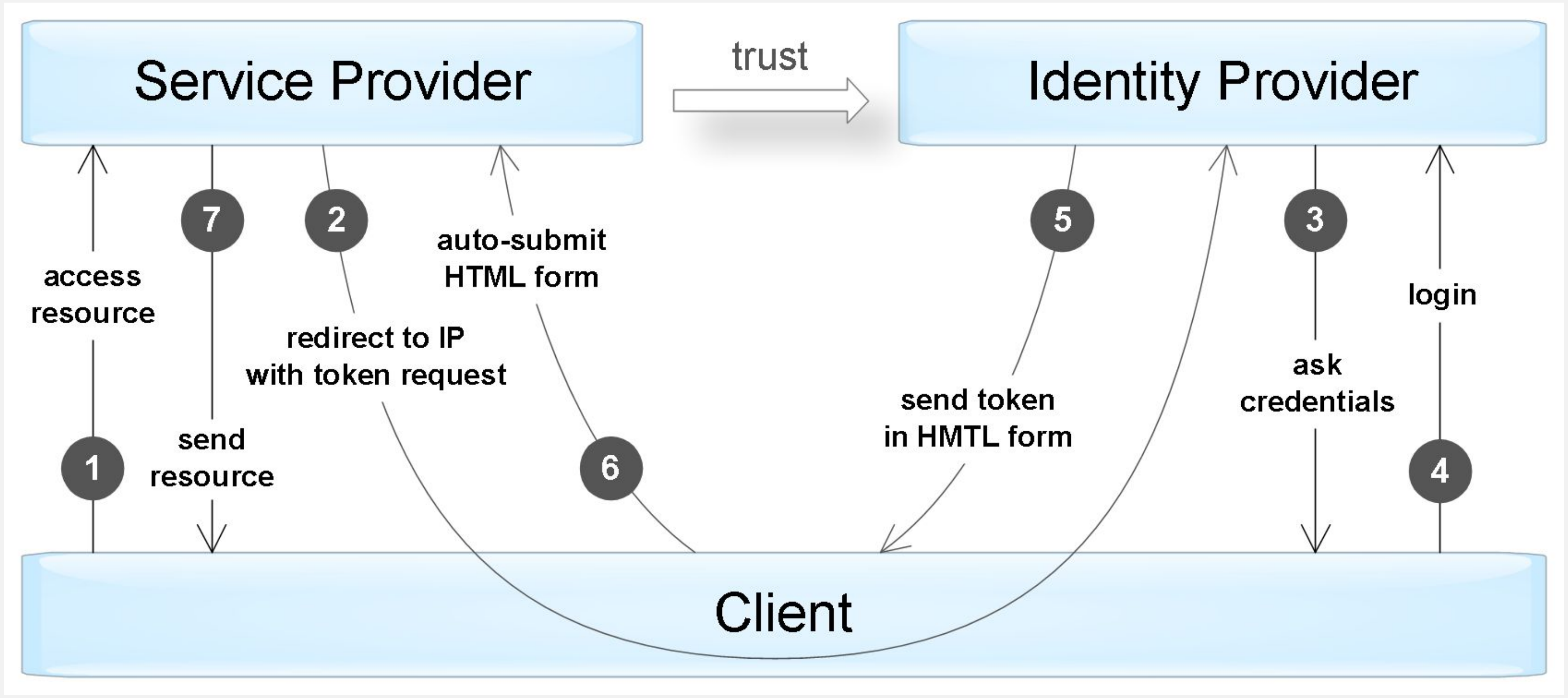
```
Digest username="Atif", realm="MADAM", qop="auth",
algorithm="MD5", uri="/madam/Download.ashx",
nonce="MS8zMS8yMDA2IDI6MzY6MTQgUE0", nc=00000002,
cnonce="5698a543e521c0fefb03abed19431564",
response="785d575418208a98f729cee48a822dd8"
```

DIGIST STRATEGY

CODE SNIPPET

```
passport.use(new DigestStrategy({ qop: 'auth' },
  function(username, done) {
    User.findOne({ username: username }, function (err, user) {
      if (err) { return done(err); }
      if (!user) { return done(null, false); }
      return done(null, user, user.password);
    });
  },  function(params, done) {
    done(null, true)
  }
));
```

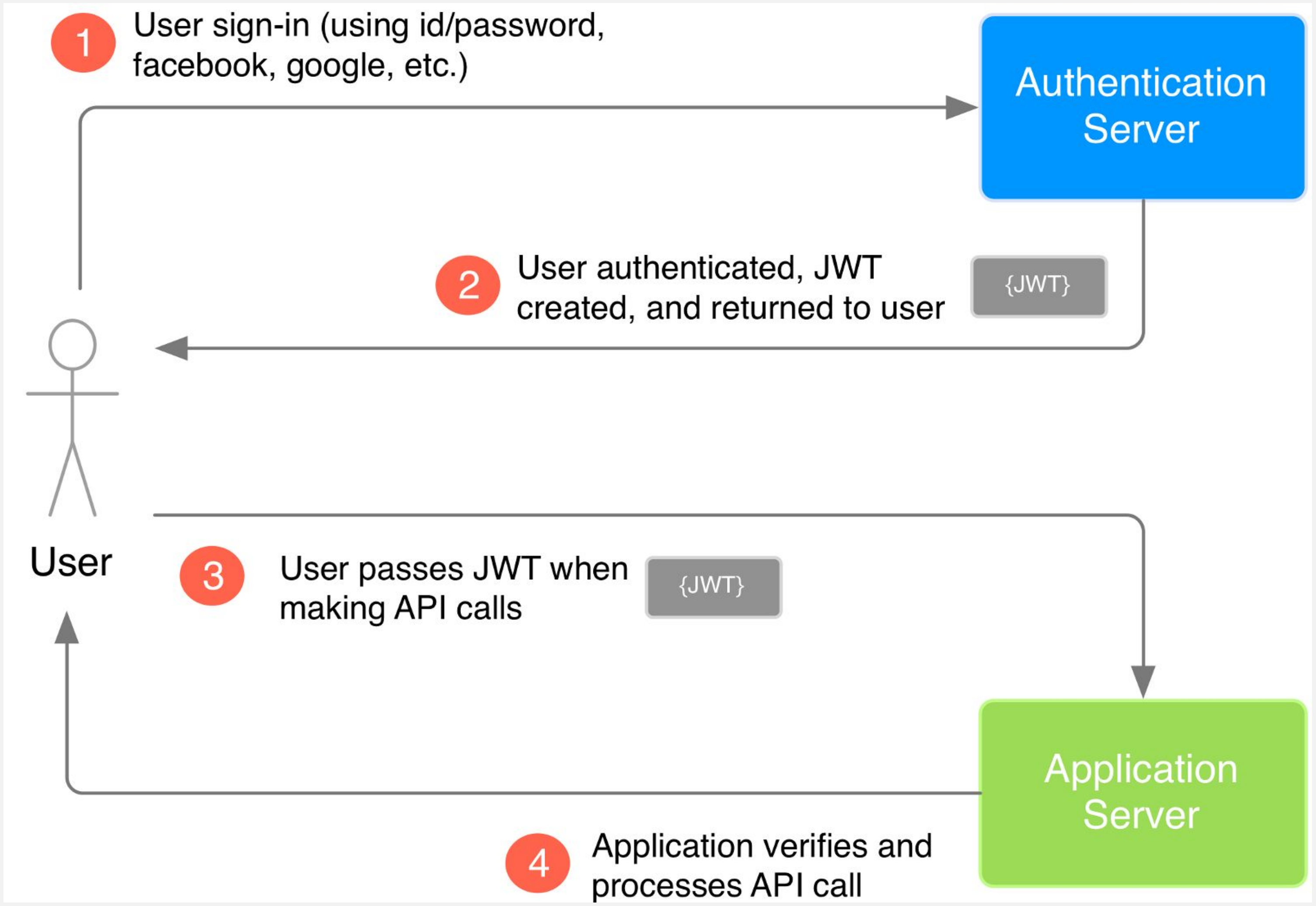
TOKEN AUTHENTICATION



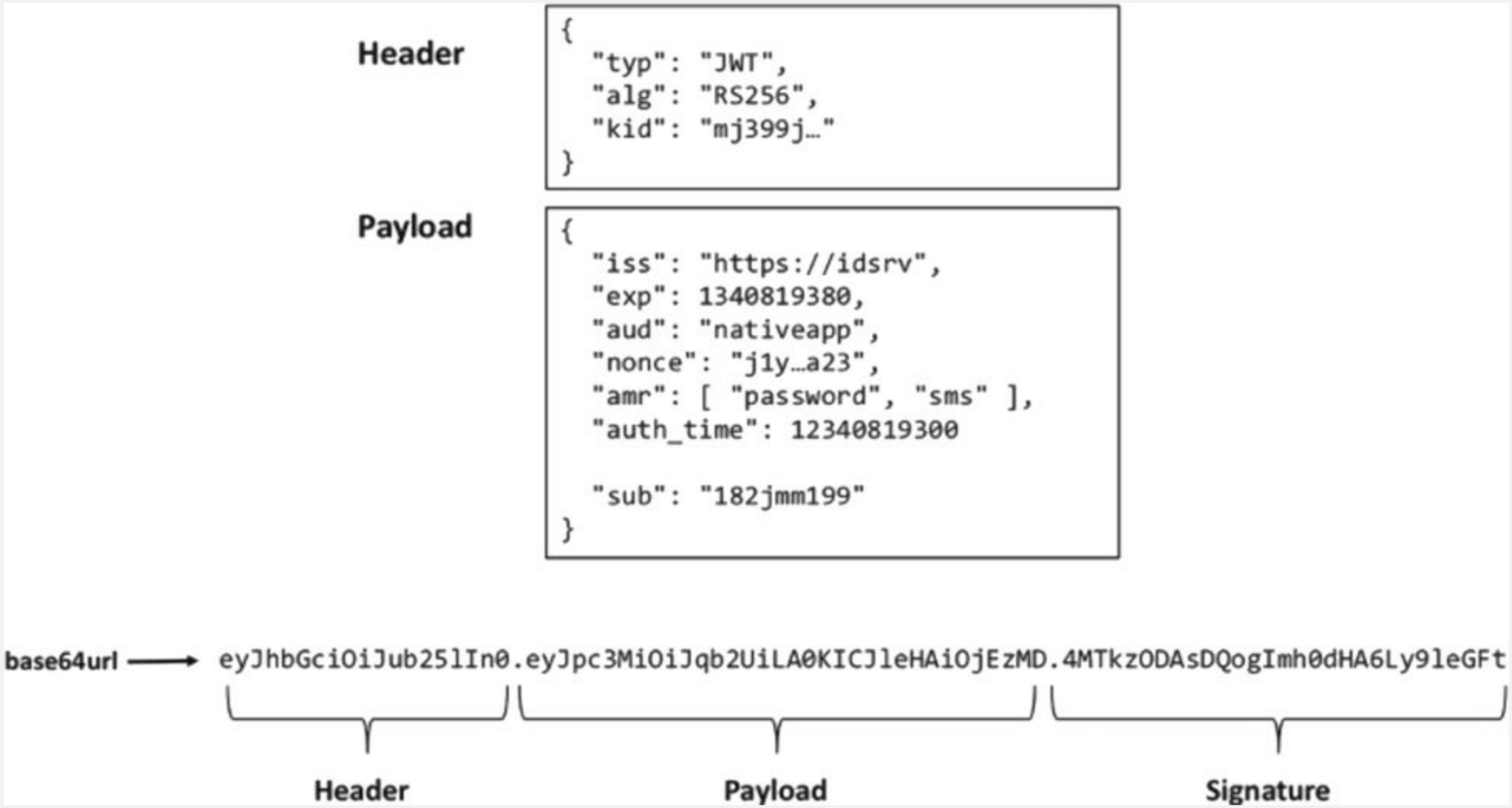
OAUTH2 & OPENID TERMINOLOGY

1. OpenID Connect Provider (OP)
2. Client
3. User
4. Scope
    - Identity scopes – openid, profile, email
    - Resource scopes – various API
5. Authentication/Token Request
6. Identity Token
7. Access Token
8. Refresh Token

JWT


1
User sign-in (using id/password, facebook, google, etc.)
Authentication Server
2
User authenticated, JWT created, and returned to user
{JWT}
User
3
User passes JWT when making API calls
{JWT}
Application Server
4
Application verifies and processes API call

TOKEN STRUCTURE



Header

```
{
  "typ": "JWT",
  "alg": "RS256",
  "kid": "mj399j…"
}
```

Payload

```
{
  "iss": "https://idsrv",
  "exp": 1340819380,
  "aud": "nativeapp",
  "nonce": "j1y…a23",
  "amr": [ "password", "sms" ],
  "auth_time": 12340819300

  "sub": "182jmm199"
}
```

base64url $\longrightarrow$ `eyJhbGciOiJub25lIn0.eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMD.4MTkzODAsDQogImh0dHA6Ly9leGFt`

GOOGLE OAUTH 2.0

CODE SNIPPET

```
passport.use(new GoogleStrategy({
    clientID: GOOGLE_CLIENT_ID,
    clientSecret: GOOGLE_CLIENT_SECRET,
    callbackURL: "http://www.example.com/auth/google/callback"
  },
  function (accessToken, refreshToken, profile, done) {
    User.findOrCreate({googleId: profile.id}, function (err, user) {
      return done(err, user);
    });
  }
));
```

GOOGLE OAUTH 2.0

CODE SNIPPET

```
passport.use(new GoogleStrategy({
    clientID: GOOGLE_CLIENT_ID,
    clientSecret: GOOGLE_CLIENT_SECRET,
    callbackURL: "http://www.example.com/auth/google/callback"
  },
  function (accessToken, refreshToken, profile, done) {
    User.findOrCreate({googleId: profile.id}, function (err, user) {
      return done(err, user);
    });
  }
));
```

JWT GENERATION

CODE SNIPPET

```
const generateToken = (id) => {
  const token = jwt.sign(
    {
      id,
      exp: Math.floor(Date.now() / 1000) +
parseInt(configAuth.JWT.live)
    },
    configAuth.JWT.secret);
  return {token: "bearer " + token}
};
```

JWT AUTH

CODE SNIPPET

```
const opts = {
  jwtFromRequest:
ExtractJwt.fromAuthHeaderAsBearerToken(),
  secretOrKey: configAuth.JWT.secret
};
passport.use(new JwtStrategy(opts, async
(jwt_payload, done) => {
  try {
    const user = await
User.findById(jwt_payload.id);
    if (user) {
      return done(null, user);
    }
```

JWT AUTH

CODE SNIPPET

```
router.get('/profile',
passport.authenticate('jwt', { session: false }),
async (req, res) => {
  res.json(req.user);
});
```

PROFILE STRUCTURE

1. provider {String}
2. id {String}
3. displayName {String}
4. name {Object}
5. familyName {String}
6. givenName {String}
7. middleName {String}
8. emails {Array} [n]
9. value {String}
10. type {String}
11. photos {Array} [n]
12. value {String}
13. The URL of the image.

LOG IN

CODE SNIPPET

```
req.login(user, function (err) {
  if (err) {
    return next(err);
  }
  return res.redirect('/users/' + req.user.username);
});
```

LOG OUT

CODE SNIPPET

```
app.get('/logout', function (req, res) {
  req.logout();
  res.redirect('/');
});
```

AUTHORIZE

CODE SNIPPET

```
app.get('/connect/twitter',
  passport.authorize('twitter-authz', { failureRedirect: '/account' }),
  function(req, res) {
    var user = req.user;
    var account = req.account;
    account.userId = user.id;
    account.save(function(err) {
      if (err) { return self.error(err); }
      self.redirect('/');
    });
  });
```

AUTHORIZE CONFIGURATION

CODE SNIPPET

```
passport.use('twitter-authz', new TwitterStrategy({...},
function(token, tokenSecret, profile, done) {
  Account.findOne({ domain: 'twitter.com', uid: profile.id },
    function(err, account) {
      if (err) { return done(err); }
      if (account) { return done(null, account); }
      var account = createTwitterAccount(token, tokenSecret);
      return done(null, account);
  });
}
));
```

AUTHORIZE CONFIGURATION

CODE SNIPPET

```
passport.use(new TwitterStrategy({
  …
  passReqToCallback: true
},
function(req, token, tokenSecret, profile, done) {
  if (!req.user) {
    // Not logged-in. Authenticate based on Twitter account.
  } else {
    // Logged in. Associate Twitter account with user. }
  }
));
```

USEFUL LINKS

- http://www.passportjs.org
- https://jwt.io/introduction/
- https://github.com/iTechJ/passport-js
- https://developers.google.com/identity/protocols/OAuth2
- https://www.npmjs.com/package/oauth2-server
- https://github.com/iTechJ/passport-js

Any questions?

iTechArt