Списание „Математика, компютърни науки и образование“ Том 6, кн. 1, 2023
Journal of Mathematics, Computer Science and Education Vol. 6, No. 1, 2023

DOI: 10.54664/XRYI9517

ИНТЕГРАЦИЯ НА IDENTITYSERVER С ASP.NET CORE

Пламенна Петрова, Силвия Върбанова

IDENTITYSERVER INTEGRATION WITH ASP.NET CORE

Plamenna Petrova, Silviya Varbanova

Abstract: *This report explores the features of Duende IdentityServer, also known as IdentityServer, which is the most widely used OpenID Connect and OAuth framework for ASP.NET Core. It allows an ASP.NET Core application to serve as an OpenID Provider and OAuth authorization server, offering both Single Sign-On (SSO) and API security as built-in features. IdentityServer takes care of the protocol support, but user authentication options can be customized by the software developer, as well.*

Keywords: *IdentityServer; OpenID Connect; OAuth; ASP.NET Core; Single Sign-On (SSO); CAT pattern; tokens; Security Token Service (STS); SPA client; API Resources.*

ВЪВЕДЕНИЕ

За разработчиците и ИТ специалистите разрешаването на проблема със защитата на данните на една работна система започва с избора на стандарт за внедряване, за да се гарантира цялостната сигурност на организацията. Водещата опция за интегриране за .NET платформата е Duende IdentityServer. Duende IdentityServer е фреймуърк, базиран на протоколите OpenID Connect и OAuth 2.0, който предоставя набор от услуги и мидълуери за ASP.NET Core приложения [10]. Фреймуъркът поддържа множество протоколни потоци или типове разрешения, разпознаваеми в документацията му като Grant Types, които представляват методи за получаване на уникални идентификатори за достъп до ресурси на обекти, без да се разкриват идентификационни данни. Автентикацията е необходима, когато дадено приложение изисква идентифицирането на потребителите си. Обикновено приложният софтуер управлява данни от името на потребителя и трябва да осигури достъп само до разрешени ресурси. В тази връзка OpenID Connect е успял да се наложи като най-новия и най-широко разпространения протокол за автентикация, притежаващ най-много потенциал за внедряване на разработван и тестван софтуерен продукт. Приложенията комуникират с приложно-програмни интерфейси[1], привеждайки своя форма за идентичност или делегирайки самоличността на потребителя. В някои случаи е нужно и двата метода да бъдат комбинирани, което е изпълнено като задача от OAuth 2.0, който добавя OpenID Connect като свое разширение. В същината на OAuth 2.0 стои правоспособността на приложния софтуер да поис-

[1] Приложно-програмният интерфейс (на английски: application programming interface, API) е интерфейсът на изходния код, който операционната система или нейните библиотеки от ниско ниво предлагат за поддръжката на заявките от приложния софтуер или компютърните програми

ка токени за достъп от защитена токен услуга – Security Token Service / STS и които да използва при комуникацията си с приложно-програмни интерфейси. Този вид делегация намалява нивата на сложност в процеса на интеграция, тъй като позволява централизирането на автентикацията и оторизацията[2] [9]. Duende IdentityServer имплементира гореспоменатите протоколи и е високо оптимизиран в повишаването на сигурността при модерните мобилни, десктоп и уеб приложения, давайки пълен контрол и възможности за работа с потребителския интерфейс, потребителското изживяване, персонализирани бизнес логика и данни [8].

ИЗЛОЖЕНИЕ

През 2021 година от четвъртата версия на IdentityServer се превключва към комерсиално лицензиране, след което фреймуъркът започва да бъде адресиран като Duende IdentityServer. Duende IdentityServer продължава да бъде система за поддръжка на операции - Operational Support System (OSS)[3], но за получаването на лиценз е нужно повечето организации да закупят такъв от разработчика Duende Software. IdentityServer4 бива поддържан през целия жизнен цикъл на .NET Core 3.1, който привършва през декември 2022 година, след което се препоръчва инсталирането на актуални NuGet пакети[4], асоциирани с версиите на Duende IdentityServer [1].

При интеграцията на IdentityServer с ASP.NET Core много често участва архитектурният шаблон CAT. Шаблонът CAT бива изграден от три модулни услуги: Clients (клиенти), API Resources (ресурси на приложно-програмни интерфейси) и Security Token Service / STS (защитена токен услуга), което способства за реализирането на принципа за разделяне на отговорности, съобразно който се постига минимално припокриване между функциите на отделните съставни единици на разработвания приложен софтуер (фиг. 1). За да се гарантира, че едно приложение е ефективно и правата на потребителите са защитени, към всяка модулна услуга се асоциират собствени правомощия и отговорности, които включват също и взаимодействието с останалите модулни услуги [6]. По-подробно се разграничава между следните роли на услугите, съставляващи шаблона CAT:

1. Clients (клиенти) – могат да бъдат десктоп, уеб и мобилни приложения

2. API Resources (ресурси на приложно-програмни интерфейси) – с помощта на тази услуга се доставят JSON[5] данни посредством REST API[6], GraphQL[7] и др.

3. Security Token Service / STS (защитена токен услуга) – генерира JWT[8] токени и улеснява метода за автентикация на единичното влизане – Single-Sign On (SSO) чрез използването на системи, които създават, съхраняват и управляват цифрови идентичности, наречени още доставчици на идентичност като Azure AD, Okta и др. или посредством начини за съхранение на локални акаунти. В същността на метода за автентикация на единичното влизане е съсредоточена схема за удостоверяване, позволяваща на потребителя да влезе с един идентификатор в някоя от няколко свързани, но независими софтуерни системи [5].

---

[2] Трябва да се направи разграничение между понятията автентикация (на английски: authentication), която в компютърната сигурност представлява установяване на самоличността на даден потребител и оторизация (на английски: authorization) – която дефинира действията, които даден потребител е упълномощен да извърши

[3] Компютърна система, използвана от доставчиците на телекомуникационни услуги за управление на техните мрежи

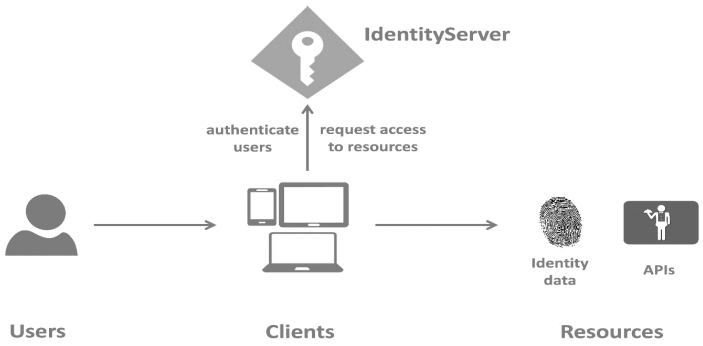
[4] Софтуер, написан с помощта на .NET framework

[5] JSON, или JavaScript Object Notation, е текстово базиран отворен стандарт, създаден за човешки четим обмен на данни. Произлиза от скриптовия език JavaScript, за да представя прости структури от данни и асоциативни масиви, наречени обекти

[6] REST API (известен също като RESTful API) е вид приложно-програмен интерфейс, който отговаря на принципите на проектиране на архитектурния стил REST

[7] GraphQL е език за заявки и манипулиране на данни с отворен код за приложно-програмни интерфейси

[8] JSON Web Token (JWT) е отворен стандарт, който дефинира компактен и самостоятелен начин за сигурно предаване на информация като JSON обект



Фигура 1. Илюстрация на участието на шаблона CAT при интеграцията на IdentityServer

IdentityServer работи с JWT токени, чиято технология е едновременно силно ефективна и опростена при автентикация на приложно-програмни интерфейси и оторизация от тип сървър към сървър. JWT токените гарантират собственост върху данните, но не и криптиране, от което следва, че JSON данните, съхранени в един JWT токен могат да бъдат видяни от всекиго, който прихване токена, тъй като той е просто сериализиран, а не криптиран, което означава, че JWT токените не правят приложно-програмните интерфейси неуязвими [3]. Поради тази причина към тях бива обединен и допълнителен слой протекция, за да се защити целият мрежови трафик с HTTPS връзка.

Ключовите особености и отговорности на IdentityServer са:

- протекция на ресурси
- автентикация на потребители със средствата на начин за локално съхранение на акаунти или външни доставчици на идентичност
- управляване на сесии[9] и реализация на метода за автентикация на единичното влизане - Single Sign-On (SSO)
- управление и автентикиране на клиенти
- издаване на токени за идентичност и токени за достъп на клиентите
- валидиране на токени

При конфигурацията на фреймуърка трябва да бъдат дефинирани основно потребители, клиенти и ресурси. В контекста на IdentityServer потребителят представлява човек, използващ регистриран клиент за достъп до ресурси [2]. Позовавайки се на фреймуърка ASP.NET Identity, IdentityServer предоставя готовия клас TestUser, с който могат да бъдат зададени идентификационни данни за тестови потребители, които ще участват във формирането на JWT токените като SubjectId – идентификатор за субект в контекста на JWT токен или определящи информационни данни - Claims като имейл и роля на потребителя:

```
        public static List<TestUser> Get()
        {
            return new List<TestUser>
    {
```

[9] Сесията е начин за съхраняване на информация (в променливи), която да се използва на множество страници. За разлика от бисквитките, при сесиите информацията не се съхранява на компютъра на потребителя

```
                new TestUser
                {
                    SubjectId = "5BE86359-073C-434B-AD2D-A3932222DABE",
                    Username = "JohnDoe",
                    Password = "JohnDoe123",
                    Claims = new List<Claim>
    {
                      new Claim(JwtClaimTypes.Email, "johndoe@gmail.com"),
                      new Claim(JwtClaimTypes.Role, "admin")
                     }
                    }
                };
      }
```

Клиентът е част от софтуера, който изисква токени от IdentityServer в сценариите за автентикация на потребител – заявка на токен за идентичност или за достъп до ресурс – заявка за токен за достъп. За да се осъществи изпращането на токени, клиентът трябва да бъде не само регистриран през IdentityServer, но и да бъде интегрирана подходяща библиотека от негова страна, която да служи за разпознаване на фреймуърка, при което идентификаторът за клиент трябва да съвпада в дефинираните настройки на IdentityServer както и в конфигурацията на клиентската библиотека (фиг. 2). От гледна точка на програмната логика, идваща от IdentityServer, клиентите могат да бъдат описани най-базово като:

```
            public static IEnumerable<Client> Get()
            {
                return new List<Client>
                {
                    new Client
                    {
                     ClientId = "SPAAngularClient",
                       ClientName = "Example SPA Angular Client",
                       AllowedGrantTypes = GrantTypes.Code,
                       ClientSecrets = new List<Secret>
        {
                         new Secret("SuperSecretPassword".Sha256())
                       },
                       AllowedScopes = new List<string> { "api1.read" }
                    }
                };
      }
```

```
export const environment = {
  production: false,
  defaultLanguage: 'en-US',
  supportedLanguages: ['en-US'],
  // Source code for API Project to run on localhost
  apiEndpoint: 'https://localhost:7246/api/v1',
  // settings for connection to Duende IdentityServer
  oidc: {
    issuer: 'https://localhost:44310', // running on localhost
    clientId: 'SPAAngularClient', // client id setup in Duende IdentityServer
    responseType: 'code', //code flow PKCE
    redirectUri: window.location.origin,
    postLogoutRedirectUri: window.location.origin,
    silentRefreshRedirectUri: window.location.origin + '/silent-refresh.html',
    scope: 'openid profile email roles', // Ask offline_access to support refresh token refreshes
    useSilentRefresh: true, // Needed for Code Flow to suggest using iframe-based refreshes
    silentRefreshTimeout: 5000, // For faster testing
    timeoutFactor: 0.25, // For faster testing
    sessionChecksEnabled: false,
    showDebugInformation: false, // Also requires enabling "Verbose" level in devtools
    clearHashAfterLogin: false, // https://github.com/manfredsteyer/angular-oauth2-oidc/issues/457#issuecomment-4318
    nonceStateSeparator: 'semicolon', // Real semicolon gets mangled by IdentityServer's URI encoding
  }
```

Фигура 2. Дефиниране на настройки за комуникация с протокола OpenID Connect в едностранично клиентско приложение – Single Page Application (SPA), изградено чрез средствата на фронтенд фреймуърка Angular в среда за разработка

Избраната опция Code от изброимия тип GrantTypes за AllowedGrantTypes както и пропъртито responseType със стойност ‘code’ от вложения обект oidc за обекта environment съобразно посочените примери са свързани със задаването на типа разрешение Authorization Code, според заложените правила на който конкретно използваният код за оторизация има временна природа и ще бъде обменен за токен на достъп. Самият код се получава от оторизационния сървър, където потребителят може да прегледа информацията, поискана от клиента и да одобри или отхвърли заявката му [7]. В случая на представеното едностранично приложение логинът се извършва на база на автентикационния процес Proof Key For Code Exchange (PKCE), който утилизира оторизационния фреймуърк OAuth 2.0 за автентикация на потребители. В рамките на този процес, потребителят първо се логва към доставчик на идентичност заедно със своите данни за идентификация. Когато автентикацията е финализирана успешно, доставчикът за идентичност генерира конкретен код за оторизация, който бива изпратен към клиентското приложение. Клиентското приложение от своя страна изпраща заявка към оторизационния сървър, като включва генерирания код за оторизация и негов верификатор. Кодовият верификатор е случайно генериран символен низ, който бива хеширан от еднопосочен хеширащ алгоритъм като SHA256 и се включва в заявката за оторизация [4]. Оторизационният сървър впоследствие извършва проверка на кода за оторизация както и на верификатора му, и ако те съвпадат, на клиентското приложение се издават токен за достъп и токен за опресняване. Токенът за достъп намира употреба при автентикацията на потребителя в последващи заявки, докато токенът за опресняване се поисква за създаване на нов токен за достъп, когато приближи изтичането на оригиналния действащ такъв. Автентикационният процес Proof Key For Code Exchange (PKCE), разглеждан като поток, надгражда с допълнителен слой на сигурност в сравнение с традиционния оторизационен поток OAuth 2.0, като предотвратява някои възможни типове атаки като прихващане на оторизационния код и атаки за повторно въпроизвеждане[10]. В цялостност логин подходът, базиран на Proof Key For Code Exchange (PKCE), е защитен и много използван процес за автентикация в модерните уеб приложения и помага в защитата срещу изтичането на идентификационните данни на потребителя и друг вид сензитивна информация. Съществуват и други типове разрешения като Client Credentials, който също е широко приложим и

[10] Атаката с повторно възпроизвеждане е форма на мрежова атака, при която валидното предаване на данни се повтаря злонамерено или се забавя. Това се извършва или от инициатора, или от противник, който прихваща данните и ги предава повторно

се използва от клиенти за получване на токени за достъп извън контекста на потребителя и следователно главната му функция е съсредоточена върху достъпването на ресурси в пряка свързаност със самите клиенти отколкото върху достъпването на ресурсите на потребителя [7].

Обхватите – scopes съставят правата на едно клиентско приложение. В IdentityServer те са обикновено моделирани като ресурси, поделящи се на два спектъра идентичност и API. Един ресурс за идентичност – IdentityResource дава възможност за моделирането на обхват, който ще позволи на клиентското приложение да разгледа подмножеството на определящите информационни данни на потребителя. Например обхватът profile способства за разпознаването на информация като потребителско име и дата на раждане. API ресурсите - ApiResources, от друга страна позволяват моделирането на достъпа до изцяло защитен ресурс или приложно-програмен интерфейс с индивидуални нива за достъп, до които клиентсктото приложение може да достигне:

```
public static IEnumerable<IdentityResource> GetIdentityResources()
 {
    return new[]
    {
           new IdentityResources.OpenId(),
           new IdentityResources.Profile(),
           new IdentityResources.Email(),
           new IdentityResource
           {
               Name = "role",
               UserClaims = new List<string> { "role" }
           }
    };
 }



public static IEnumerable<ApiResource> GetApiResources()
{
    return new[]
    {
         new ApiResource
          {
           Name = "api1",
           DisplayName = "API #1",
           Description = "Allow the application to access API #1 on
your behalf",
           Scopes = new List<string> {"api1.read", "api1.write"},
           ApiSecrets = new List<Secret> {new Secret(„ScopeSecret".
Sha256())},
           UserClaims = new List<string> {"role"}
      }
    };
}


      public static IEnumerable<ApiScope> GetApiScopes()
```

```
{
    return new[]
    {
        new ApiScope("api1.read", "Read Access to API #1"),
        new ApiScope("api1.write", "Write Access to API #1")
    };
}
```

Първите три вида ресурси за идентичност OpenId, Profile и Email представляват стандартни OpenID Connect обхвати, поддържани от IdentityServer. За конкретния пример обхватът email се отнася до определящите информационни данни за потребителя – стойност на имейла и текущото състояние за верификацията му. В допълнение ресурсът за идентичност role определя ролята на автентикирания потребител, която също спада към неговите определящи информационни данни, а обхватът OpenId винаги е нужен за получаване на токен за идентичност от протокола OpenID Connect. API обхватът – ApiScope е индивидуално ниво на оторизация към конкретния приложно-програмен интерфейс, към който клиентското приложение изпраща заявки. Трябва да се обърне внимание и на факта, че обхватите определят какво потребителят оторизира клиентското приложение да направи от тяхно име, което не означава, че той има разрешение да извърши дадено действие, понеже протоколът OAuth не се явява доставчик на оторизация на потребителско ниво, а само на клиентско.

ЗАКЛЮЧЕНИЕ

При Duende IdentityServer адаптирането на собствено дефирани бизнес правила и персонализирането на работните потоци е водещо, което го прави най-гъвкавия и съвместим със стандартите на протоколите OpenID Connect и OAuth 2.0 фреймуърк за ASP.NET Core. Ето защо е предоставена свобода за избор по преценка на софтуерните разработчици за подходящ фреймуърк за потребителски интерфейс или технология при интеграцията на IdentityServer. Липсата на принуда за обвързаност със специфична хостинг среда, принадлежаща към точно определена база данни или географски регион също е важен фактор за популяризирането на фреймуърка. IdentityServer може да работи на всички операционни системи и продукти за виртуализация на ниво ОС като Docker и Kubernetes. Отвореността на сорс кода му в GitHub допринася за бързия процес на развитие, наблюдаван при него, тъй като добрите практики като публично следене на проблеми и заявки за обединение на промени извеждат IdentityServer извън капсулацията на една частна среда на разработка и стимулират по-бързото отстраняване на неизправности, разбиране и учене в реални работни сценарии.

ЛИТЕРАТУРА

[1] Scott Brady. 2021. Getting Started with IdentityServer4 and Duende IdentityServer. (Август, 2021). Извлечено на 14.05.2023 г. от https://www.scottbrady91.com/identity-server/getting-started-with-identityserver-4

[2] Kavindu Dodanduwa, Ishara Kaluthanthri. 2018. Role of Trust in OAuth 2.0 and OpenID Connect (Август, 2018), 1-5.

[3] László Viktor Jánoky, J. Levendovszky, Péter Ekler. 2018. An analysis on the revoking mechanisms for JSON Web Tokens. International Journal of Distributed Sensor Networks 14(9):155014771880153 (Септември, 2018), 1-10. https://doi.org/10.1177/1550147718801535

[4] Fuji Nguyen. 2023. Angular 15 & Net 7: Login and Token Refresh Integration with Duende IdentityServer | Tutorial 10. (Март, 2023). Извлечено на 14.05.2023 г. от https://medium.com/scrum-and-coke/angular-15-and-net-7-login-and-token-refresh-with-duende-identityserver-83dc38aa8d5

[5] Jonas Primbs, Michael Menth. 2023. OIDC^2$: Open Identity Certification with OpenID Connect (Юли, 2023), 1-27.

[6] Husain Salah. 2023. I - Understanding CAT and Identity Server. (Януари, 2023). Извлечено на 14.05.2023 г. от https://husainsalah.hashnode.dev/i-understanding-cat-and-identity-server#heading-what-is-cat

[7] Jaimandeep Singh, Naveen Kumar Chaudhary. 2023. Unified Singular Protocol Flow for OAuth (USPFO) Ecosystem (Януари, 2023), 1-10. https://arxiv.org/abs/2301.12496\

[8] Duende Software. 2021. IdentityServer Product Overview. Извлечено от https://duendesoftware.com/products/identityserver

[9] Plain Concepts. 2021. Duende IdentityServer. Извлечено от https://www.plainconcepts.com/duende-identity-server-partner/

[10] Kravets A.V. , Понамарьов Ігор Володимирович. 2021. ІНТЕГРАЦІЯ IDENTITY SERVER ФРЕЙМВОРКУ ЯК СТОРОННЬОГО ПОСТАЧАЛЬНИКА АВТЕНТИФІКАЦІЇ У ASP.NET CORE ДОДАТОК, 66. VI Всеукраїнська науково-практична конференція «ПЕРСПЕКТИВНІ НАПРЯМКИ СУЧАСНОЇ ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ І КОМП'ЮТЕРНИХ СИСТЕМ» MEICS-2021 (Ноември, 2021), 66-68 [Kravets A.V. , Ponamaryov Igor Volodimirovich. 2021. INTEGRATsIYa IDENTITY SERVER FREYMVORKU YaK STORONNYOGO POSTAChALYNIKA AVTENTIFIKATsIЇ U ASP.NET CORE DODATOK, 66. VI Vseukraïnsyka naukovo-praktichna konferentsiya «PERSPEKTIVNI NAPRYaMKI SUChASNOЇ ELEKTRONIKI, INFORMATsIYNIH I KOMP'YuTERNIH SISTEM» MEICS-2021 (Ноември, 2021), 66-68]

ИНФОРМАЦИЯ ЗА АВТОРИТЕ

Пламенна Петрова – студент в магистърската програма Уеб технологии и разработване на софтуер, Факултет „Математика и информатика", Великотърновски университет „Св. св. Кирил и Методий", България, Програмист, софтуерни приложения в Тремол ООД, e-mail: plamennavp@abv.bg

Силвия Върбанова – главен асистент, доктор, Факултет „Математика и информатика", Великотърновски университет „Св. св. Кирил и Методий", e-mail: s.varbanova@ts.uni-vt.bg

ABOUT THE AUTHORS

Plamenna Petrova – Master's degree student in Web Technologies and Software Development at the Faculty of Mathematics and Informatics, St. Cyril and St. Methodius University of Veliko Tarnovo, Bulgaria; Software Developer at Tremol Ltd, e-mail: plamennavp@abv.bg

Silvia Varbanova – Senior Lecturer, PhD, Faculty of Mathematics and Informatics, St. Cyril and St. Methodius University of Veliko Tarnovo, e-mail: s.varbanova@ts.uni-vt.bg