# Github bağlantısı

<https://github.com/fawz24/LezzetYolculugu.git>

# Video bağlantısı

<https://youtu.be/Jy1eP4UzXu4>

# Analiz

## Uygulamanın yapması beklenenler

Lezzet Yolculuğu uygulaması yemek tarife paylaşımı için kullanılacaktır. Kullanıcılar yemek tarifelerini arayabilir ve/veya yemek tarifesini paylaşabilirler. Ayrıca kullanıcılar yemek tarifesi hakkında yorum yapabilirler.

## Kullanıcılar

Web uygulamasına erişebilen herkes kullanıcı olarak tanımlanır. Kullanıcılar 3 ayrı sınıfa girer:

1. Anonim kullanıcı: paylaşım (yemek tarifesi, yorum) görebilir ve kayıt ya da giriş yapabilir
2. Üye kullanıcısı: paylaşım görebildiği gibi, giriş yapabilir, paylaşım (yemek tarifesi, yorum) yapabilir, kendi paylaştığı yemek tarifelerini değiştirebilir veya silebilir, yaptığı yorumları silebilir, üyeliğini iptal edebilir
3. Uygulama yöneticisi: üye kullanıcıların yetkilendirildiği tüm haklar yanı sıra, yapılan herhangi bir paylaşımı kaldırabilir, üye kullanıcıların üyeliğini iptal edebilir, yönetici kullanıcı ekleyebilir

## Giriş verileri

* Kullanıcı bilgisi: ad, soyadı, eposta, şifre
* Yemek tarifesi
* Yemek malzemeleri ve miktarları
* Yorum

## Çıkış verileri

* Kullanıcı bilgisi: ad, soyadı
* Yemek tarifesi
* Yemek malzemeleri ve miktarları
* Yorum

## İşlenecek veriler

* Kişi: ad, soyadı, eposta, şifre
* Yemek tarifesi: ad, tarife, paylaşım tarihi, malzeme listesi
* Malzeme: ad, miktar
* Miktar: nicelik, ünite

## Varlıklar

1. Kullanıcılar

* Kullanıcıların kişisel bilgileri
* Kullanıcıların giriş bilgileri
* Kullanıcıların yetki bilgileri
* Admin kullanıcıların giriş bilgileri

1. Sistemler

* Web sunucusunun erişilebilirliği
* Veri tabanı sunucusunun erişilebilirliği
* Veri tabanından veri okuma
* Veri tabanına veri yazma

1. Web uygulaması

* Uygulama dosyaları
* Oturum bilgileri
* Veri tabanına okuma, yazma, değiştirme ve/veya silme işlemleri için erişimi
* Kullanıcının oluşturulması/tanımlanması
* Admin kullanıcıların oluşturulması/tanımlanması

## Harici varlıklar

* Web kullanıcıları
* Üye kullanıcıları
* Admin kullanıcılar

## Harici bağımlılıklar

* Windows 10
* Dotnet Core 2.2
* IIS veya IIS Express
* Microsoft SQL Server 2014 +

## Giriş noktaları

* Ana sayfa
* Arama fonksiyonu
* Giriş sayfası
* Kayıt sayfası
* Admin kullanıcı ekleme sayfası
* Yemek tarifesini oluşturma sayfası
* Yemek tarifesini değiştirme sayfası
* Yemek tarifesini silme fonksiyonu
* Yorum ekleme fonksiyonu
* Yorum silme fonksiyonu

## Güven seviyeleri

* Anonim kullanıcı: giriş yapmamış ve uygulamayı kullanan
* Geçersiz kullanıcı: hatalı giriş bilgileri ile giriş yapmayı deneyen
* Üye: giriş yapmış normal kullanıcı
* Admin: giriş yapmış admin kullanıcı
* Veri tabanı admin: tüm yetkilere sahip veri tabanı yöneticisi
* Veri tabanı okuma yetkilisi: veri tabanından veri okuma yetkisi olan kullanıcı
* Veri tabanı yazma yetkilisi: veri tabanına veri yazma yetkisi olan kullanıcı

## Kullanım diyagramı

A close up of a map

Description automatically generated

Uygulama kullanım diyagramı

A close up of a map

Description automatically generated

Uygulama fonksiyon ilişkileri

## Veri tabanı varlık bağlantı modeli

A screenshot of a cell phone

Description automatically generated

## Sıralama diyagramı

A screenshot of a social media post

Description automatically generated

## Tehdit modeli

# Sızma testi raporu

## Broken Authentication ve SQL Injection

OWASP ZAP uygulaması ile web uygulamayı taradık. Web uygulamanın giriş sayfasında bir SQL Injection zafiyetinin bulunduğu anlaşıldı. Bir kullanıcı oluşturduktan sonra bu kullanıcı eposta adresi ile giriş yaparken şifre alanına “abc' OR 1=1;--” payloadunu girdiğimiz zaman o kullanıcı yerine geçmiş olduk. Sosyal mühendislik yöntemiyle bir kullanıcı epostasını ele geçerdiğimizde o kullanıcının yerine geçebiliriz. Giriş sayfasında bulunan bu SQL Injection zafiyeti bize kimlik doğrulama zafiyetinin olduğunu söylemektedir. Bir kullanıcının eposta adresi ele geçirildiğinde saldırgan bu kullanıcı yerine geçebilir. Üstelik ele geçirilen kullanıcı epostası bir admin kullanıcıya ait ise o kullanıcının tüm yetkileri saldırganın eline geçmektedir.

![A screenshot of a cell phone

Description automatically generated]()

## Cross site scripting

XSS zafiyetinin olup olmadığını kontrol etmek için giriş noktaları (formlar) denenebilir. Denemelerden yemek tarife oluşturma sayfasındaki tarife alanında bir html tag yerleştirildiğinde ana sayfada yerleştirilen bu html tag’taki kod tarayıcı tarafından yorumlandığı görülmektedir. Tarife oluştururken veri tabanına tarife alanında bir saldırı kod kaydedildiği anlaşılmaktadır. Sunucu kullanıcının girdiği verileri filtrelememektedir. Ana sayfa oluşturulurken veri tabanından okunan filtrelenmemiş kullanıcı girdisi eklenerek tarayıcının yorumlaması sağlanmaktadır.

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

## External redirect

OWASP ZAP uygulaması ile tarama sonucun giriş sayfasında harici yönlendirmenin olduğu görülebilir. Bu zafiyet giriş başaralı bir şekilde yapıldığı zaman ReturnUrl parametresi ile belirlenen sayfaya yönlendirme yapılabilir. Yönlendirilen sayfa saldırganın hazırladığı zararlı uygulama içerisinde yer alabilir. Kullanıcı web uygulamamızın dışına yönlendirilmektedir.

![A screenshot of a cell phone

Description automatically generated]()

# İncelenen güvenlik zafiyetlerinin tanıtımı ve nasıl giderildiği

## Broken Authentication

Bir uygulamada kimlik doğrulama ve yetkilendirme politikalarının doğru tanımlanmaması ciddi bir zafiyet oluşturmaktadır. Saldırganlar bu zafiyetten faydalandığı zaman erişmeye yetkileri olmayan fonksiyon ve verilere, uygulama kullanıcı bilgilerine (kimlik verileri, kişisel verileri, paylaştığı veriler…), dosyalara erişebilir, onları değiştirebilir ve/veya silebilir.

Bizim uygulamada yetkilendirme 3 seviye ile yapılmıştır. Anonim, Normal ve Admin yetkileri tanımlanmıştır. Her birinin veri tabanında yapacağı işlemler tanımlanmıştır. Bu yetkilendirme hem web uygulamasında hem de veri tabanında yapılmıştır.

Kimlik doğrulama kullanıcı şifresi ile yapılmaktadır. Kullanıcı parolaları açık metin olarak kaydedilmektedir. Bu kimlik doğrulama mekanizması güvenli değildir. Bunun güvenlik seviyesi arttırmak için kullanıcı parolaları SHA 256 algoritması ile şifrelenmiştir.

## SQL Injection

Kullanıcı girdileri doğrudan SQL sorgusuna yerleştirilerek yapılan sorgunun beklenilen davranışı değiştirilebilir. Bu sorgu uygulama için zararlı olmaktadır. Veri tabanındaki kontrolleri atlamak, verileri sızdırmak ve veri erişme engeli aşmak gibi kötü sonuçlar meydana gelebilmektedir.

Bizim uygulamada, kullanıcı parolaları açık metin olarak kaydedilmektedir. Dolayısıyla giriş sayfasında bir SQL Injection zafiyeti oluşmaktadır. Kullanıcı parolaları SHA 256 algoritması ile şifrelenerek ve uygulamadaki tüm SQL sorguları *parameterized SQL* ifadelerine dönüştürülerek bulunan SQL Injection zafiyeti giderilmiştir.

## Cross Site Scripting

XSS zafiyeti kullanıcı girdisini filtrelemeden, valide etmeden ve/veya *escape* etmeden doğrudan web sayfanın koduna yerleştirmekten kaynaklanmaktadır. Zararlı kodu içeren web sayfası kullanıcı tarayıcısına ulaştığında zararlı kod çalıştırılarak kullanıcı tarayıcısındaki oturum bilgileri, çerezleri, tarayıcı geçmişi ve tarayıcıda saklanmış diğer veriler çalınabilir ve/veya kullanıcı zararlı web sitelerine yönlendirilebilir.

Bizim uygulama ana sayfası kullanıcı girdisi kullanılarak oluşturulmaktadır. Netice olarak XSS zafiyeti içermektedir. Zafiyeti kaldırmak için kullanıcı girdisi valide ve *escape* edilmiştir.