Cross Site Request Forgery (CSRF)/Change Password

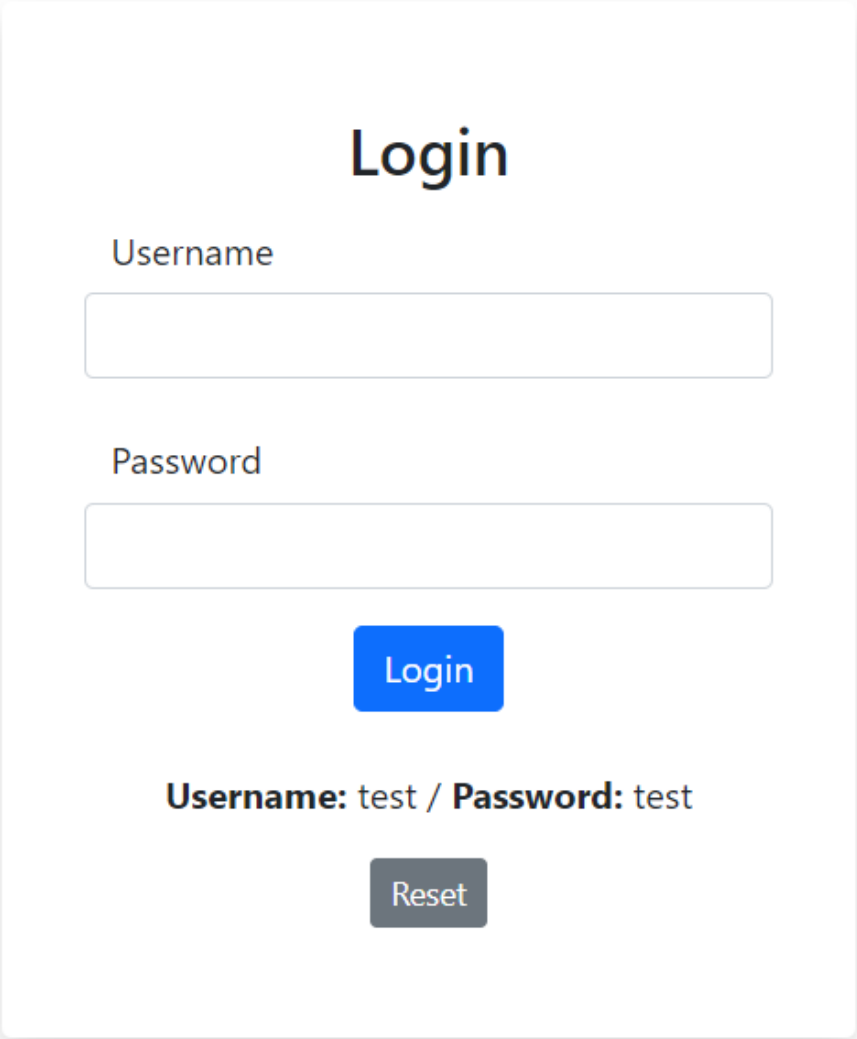
Herkese selamlar bu yazımda Hackviser üzerinde bulunan Web security lablarından biri olan **Cross Site Request Forgery (CSRF)/Change Password** çözümünü anlatacağım.

CSRF (Cross-Site Request Forgery) açığının ne olduğunu kısaca hatırlayalım:

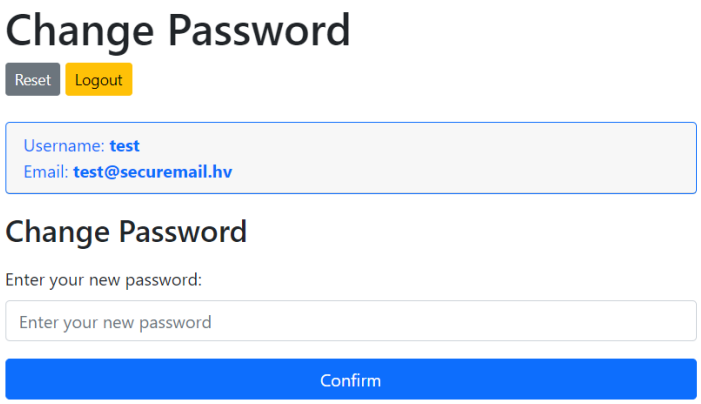
CSRF açığı, bir kullanıcının web uygulamasında oturum açmış olduğu oturumu kullanarak, kullanıcının bilgisi dışında ve isteği olmadan işlemler yapılmasına olanak tanır. Bu açık, web sitelerinin kullanıcıdan gelen talepleri doğrulamak için yeterli güvenlik önlemleri almamasından kaynaklanır. Saldırgan, hedef kullanıcının oturumunda, saldırganın istediği işlemleri (örneğin, parola değiştirme, para transferi gibi) gerçekleştirebilir.

Sırada ise makinenin çözümü var.

Bize verilen siteye ilk gittiğimizde bizi aşağıdaki gibi bir login page karşılıyor



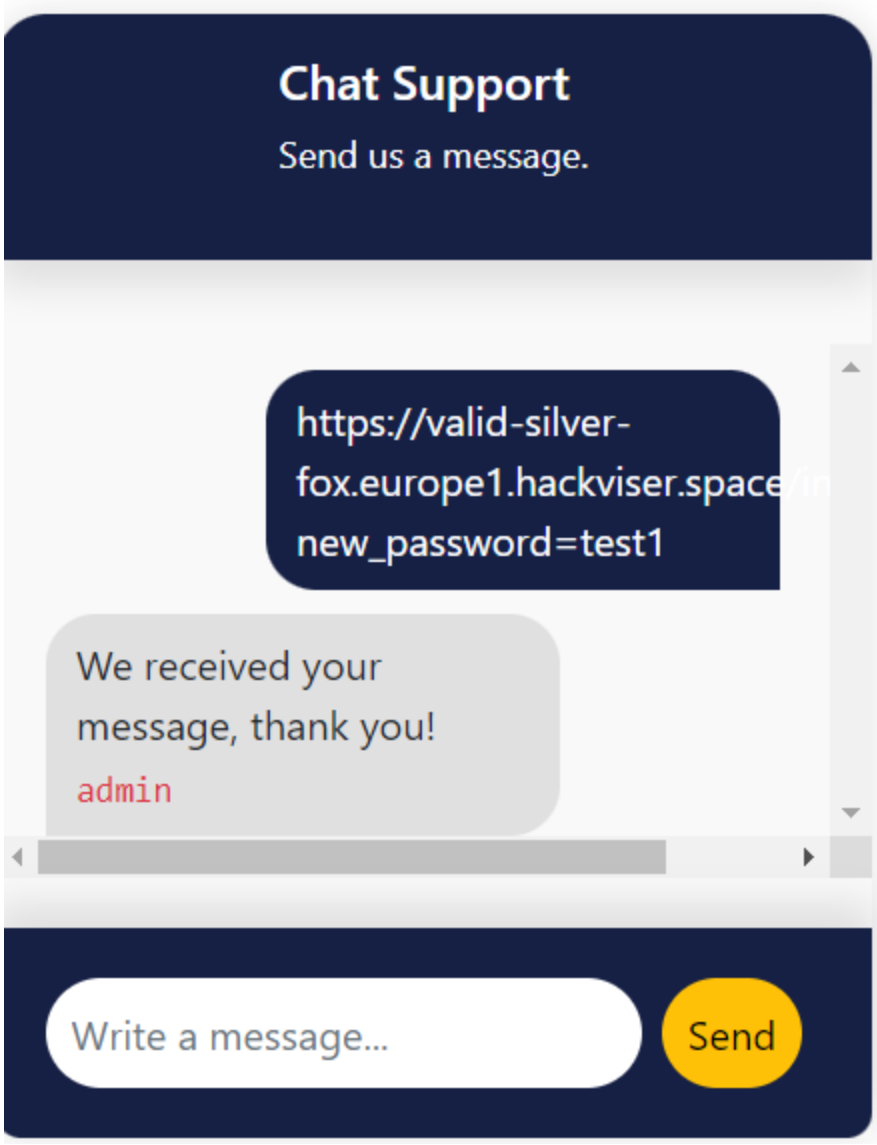
Burada verilen default bilgilerle giriş yapıyoruz.



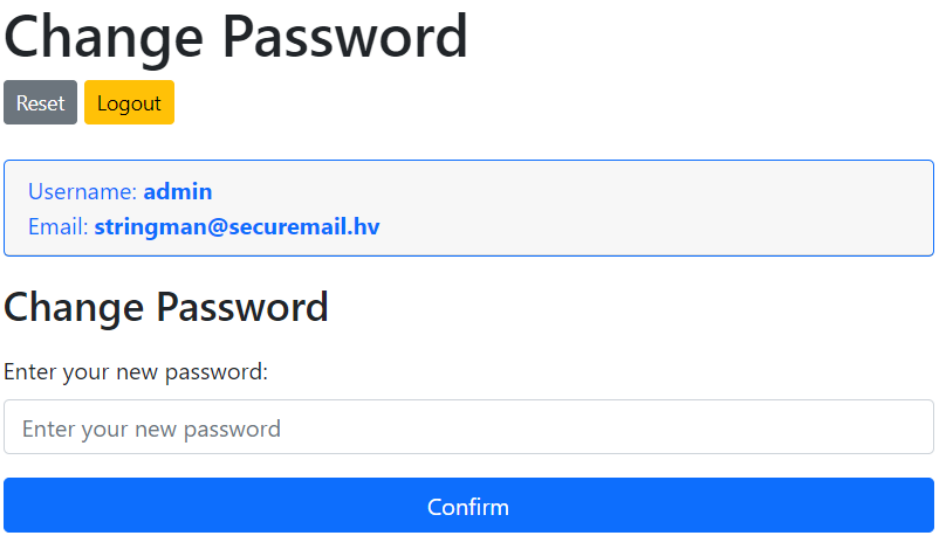
Bizi yukarıdaki gibi bir ekran karşılıyor. Şifre değiştirme kısmı ve canlı destek kısmı bizim dikkatimizi çekiyor.

index.php?new_password=test1

Şifre değiştirme işlemi yaptığımız zaman yukarıdaki gibi url uzantısı ekleniyor. Aklıma bu url yi canlı desteğe atarsam ne olur gibi bir düşünce geliyor.



Ardından test kullanıcısından çıkış yapıp değiştirdiğim şifre bilgisi ile giriş yapmayı deniyorum.



Başarılı bir şekilde admin hesabına giriş yapabildik...

CSRF (Cross-Site Request Forgery) açığını kapatmak için alınabilecek önlemler şunlardır:

1. **CSRF Token Kullanımı:** Her işlem için benzersiz bir CSRF token üretilmeli ve bu token sunucu tarafından doğrulanmalıdır. Token, form verilerine veya URL parametrelerine eklenir ve her istekte sunucu tarafında kontrol edilerek işlem yapılır. Böylece, sadece yetkili ve güvenilir isteklerin işlenmesi sağlanır.
2. **SameSite Cookie Özelliği:** Çerezlerin `SameSite` politikası kullanılarak sadece aynı site üzerinden yapılan taleplerin çerezlere erişimi sağlanabilir. Bu, üçüncü taraf sitelerden gelen kötü niyetli taleplerin etkisiz hale getirilmesine yardımcı olur. Çerezler `SameSite=Strict` veya `SameSite=Lax` olarak ayarlanabilir.
3. **Doğrulama Gereksinimleri:** Hassas işlemler (şifre değiştirme, para transferi vb.) için ek doğrulama adımları eklenebilir. Örneğin, kullanıcıdan şifresini tekrar girmesi istenebilir ya da iki faktörlü kimlik doğrulama (2FA) kullanılabilir.
4. **GET Taleplerinde Hassas İşlemlerden Kaçınma:** CSRF saldırılarının çoğu GET istekleri ile yapılabildiği için, hassas işlemler (örneğin, veri değiştirme, para

gönderme) GET yerine POST gibi daha güvenli HTTP metodları ile yapılmalıdır. GET taleplerinde veri değişikliği yapılmamalıdır.

Bu önlemlerden birkaçını uygulayarak CSRF zafiyetlerinin büyük oranda önüne geçilebilir. Güvenlik bilincine sahip bir geliştirme süreci ile uygulamanın güvenliği arttırılabilir.

Umarım bu yazıyı sıkılmadan okumuşsunuzdur. Keyifli çalışmalar.