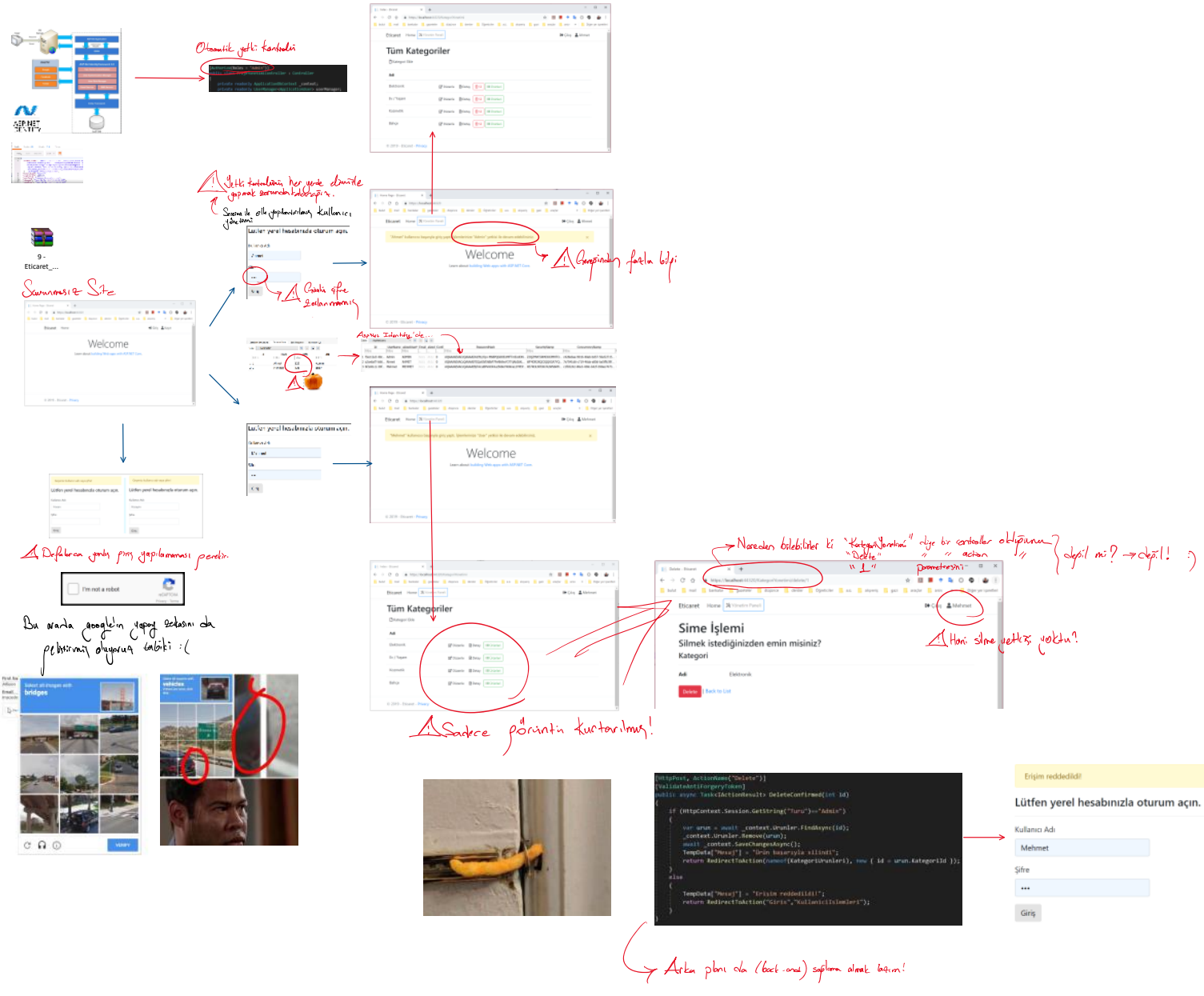
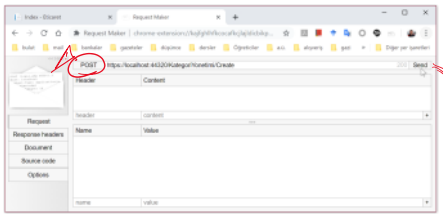
11 - Güvenlik Önlemleri

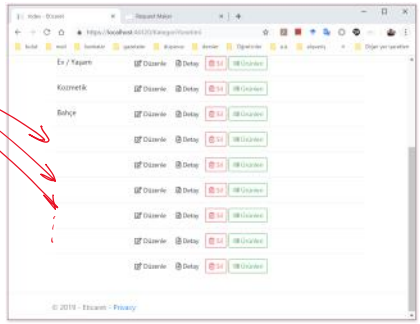
7 Mayıs 2019 Salı 04:47

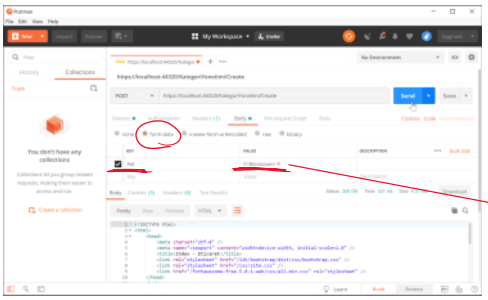
- URL saldırıları (Get, Post, PostMan, RequestMaker,AntiForgeryToken)
- Tahmini Zorlaştırmak (Guid Kullanmak)
- Kullanıcıya gereğinden fazla bilgi vermemek
- Şifrelerin Kriptolu saklanmaması
- Gidip gelen Verilerin Şifrelenmesi (Https protokolü)
- SqlInjection Saldırıları (https://www.w3schools.com/sql/sql_injection.asp)
- OverPosting Ataklar (http://www.abhijainsblog.com/2015/04/over-posting-attack-in-mvc.html)
- Dosya Yüklemeleri (İstenmeyen dosya türleri, uygunsuz resimler, büyük boyutlu dosyalar vb. yüklenmesi)
- HTML alanlarından betikler yüklenmesi
- Connection string - konfigurasyon dosyasında
- Doğrulamada istemci tarafıyla yetinmeme (Sunucu taraflı doğrulama - Model.isValidate)
- Brute-force(Kaba kuvvet saldırısı)
- CSRF/XSRF(Cross-Site Request Forgery) Saldırıları
- "man-in-the-middle" attacks
- Open Redirect Saldırıları (ReturnUrl)
- Whois Sorguları
- Environment - Product

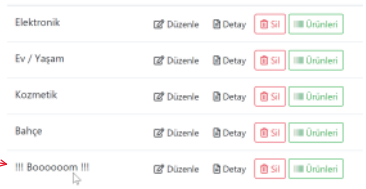



```
[HttpPost]
//[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("Id,Adi")] Kategori kategori)
{
    if (ModelState.IsValid)
    {
        _context.Add(kategori);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(kategori);
}
```

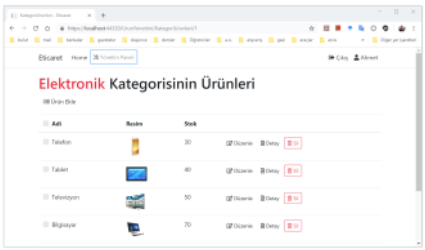




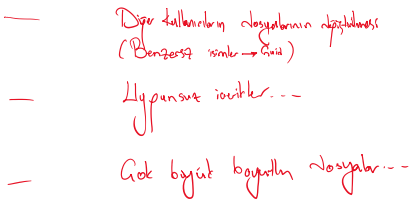


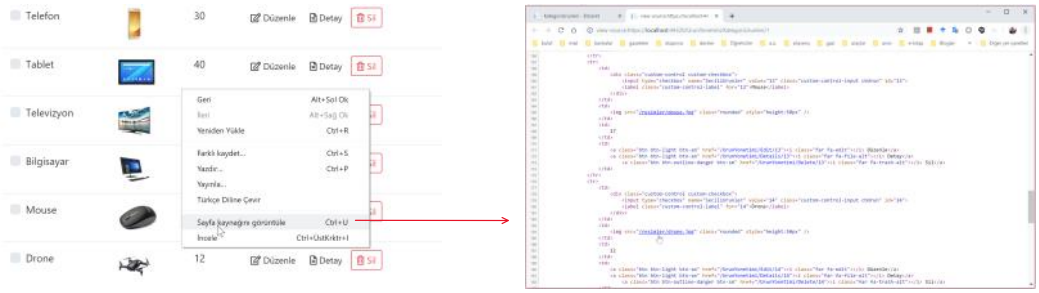


⚠ Dosya yükleme işlemlerine dikkat



— *.exe → Çok tehlikeli (Bulunduğu ortamdaki tüm dosyaları silebilir, herhangi bir portu dinlemeye alabilir, savaşı yapabilir vb...)

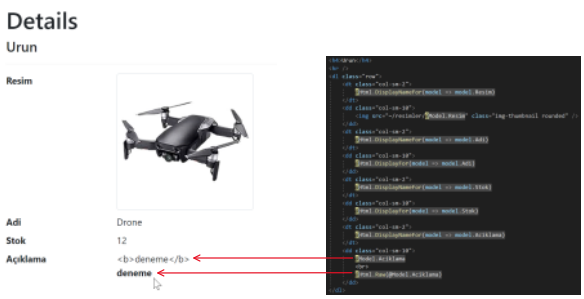


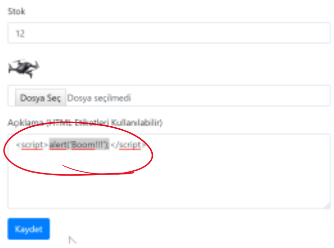


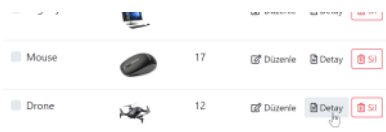
HTML alanlarına dikkat!

XSS Saldırıları

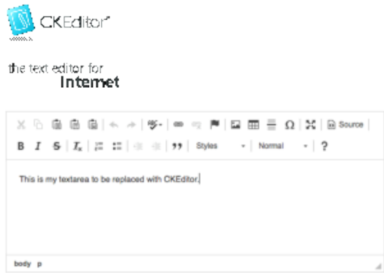
Siteler Arası Komut Dosyası Çalıştırma (XSS), bir saldırganın istemci tarafı komut dosyalarını (genellikle JavaScript) web sayfalarına yerleştirmesini sağlayan bir güvenlik açığıdır. Diğer kullanıcılar etkilenen sayfalar yüklediğinde, saldırganın komut dosyaları çalıştırılır; saldırganın çerezleri ve oturum simgelerini çalmasını sağlar, web sayfasının içeriğini DOM manipülasyonu yoluyla değiştirir veya tarayıcıyı başka bir sayfaya yönlendirir. XSS güvenlik açıkları, genellikle bir uygulama kullanıcı girişi yaptığında ve doğrulama, kodlama veya çıkış yapmadan bir sayfaya çıkardığında ortaya çıkar.

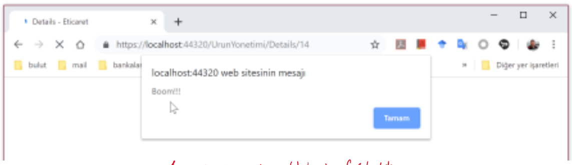






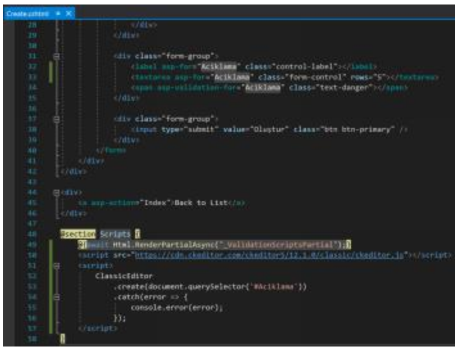
<script>alert('!!! Boom !!!');</script>

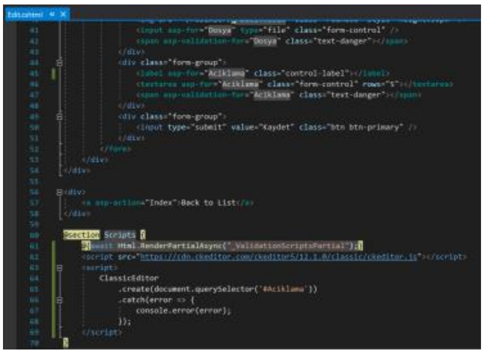


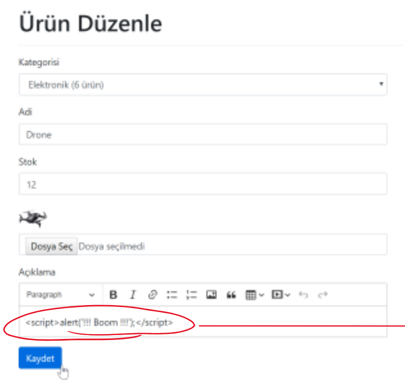


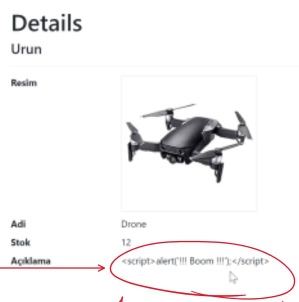
- Açık olan diğer oturum bilgilerinden faydalanabilir
- Geçmişini görüntüleme
- Kredi kartı bilgileri
- Benzer bir sahte siteye (fake) yönlendirebilir
- Reklam amaçlı bir siteye yönlendirebilir.
- Reklam amaçlı istenmeyen pencereler vb.

CK Editörü Kurmak İçin
https://ckeditor.com/docs/ckeditor5/latest/builds/guides/quick-start.html









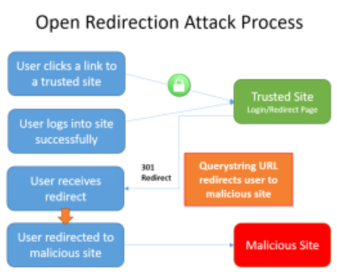
⚠ Script çalıştırılmadı

Kendi tag'lerimizi oluşturmak

<koyu> deneme </koyu>

Açık yönlendirme (open redirect) saldırısı nedir?

Web uygulamaları, kullanıcıları kimlik doğrulama gerektiren kaynaklara eriştiklerinde sık sık bir giriş sayfasına yönlendirir. Yönlendirme, tipik olarak bir `returnUrl`sorgulayıcı parametresi içerir, böylece kullanıcı başarıyla giriş yaptıktan sonra orijinal olarak istenen URL'ye geri döndürülebilir. Kullanıcı kimliği doğrulandıktan sonra, orijinal olarak istedikleri URL'ye yönlendirilir.
Hedef URL, isteğin sorgu yazısında belirtildiği için, kötü niyetli bir kullanıcı sorgu yazısına müdahale edebilir. Tahrif edilmiş bir sorgu yazma, sitenin kullanıcıyı harici, kötü niyetli bir siteye yönlendirmesine izin verebilir. Bu teknik açık yönlendirme (veya yönlendirme) saldırısı olarak adlandırılır.



Örnek bir saldırı

Kötü niyetli bir kullanıcı, kötü niyetli kullanıcının bir kullanıcının kimlik bilgilerine veya hassas bilgilerine erişmesine izin vermek amacıyla bir saldırı geliştirebilir. Saldırıya başlamak için, kötü niyetli kullanıcı, kullanıcıyı sitenizin giriş sayfasına `returnUrl`giden bir URL'yi ekleyen querystring değeri olan bir bağlantıyı tıklamaya ikna eder . Örneğin, adresindeki `contoso.com`bir giriş sayfası içeren bir uygulamayı düşünün `http://contoso.com/Account/LogOn?returnUrl=/Home/About`. Saldırı şu adımları izler:

1. Kullanıcı kötü amaçlı bir bağlantıyı tıklar `http://contoso.com/Account/LogOn?returnUrl=http://contoso1.com/Account/LogOn`(ikinci URL "contoso.com" değil , "contoso 1 .com" olur).
2. Kullanıcı başarıyla giriş yapıyor.

3. Kullanıcı (site tarafından) `http://contoso1.com/Account/LogOn`(gerçek siteye benzeyen kötü amaçlı bir site) adresine yönlendirilir .
4. Kullanıcı tekrar giriş yapar (kötü amaçlı siteye kimlik bilgilerini verir) ve tekrar gerçek siteye yönlendirilir.

Kullanıcı muhtemelen ilk giriş yapma girişimlerinin başarısız olduğuna ve ikinci girişimlerinin başarılı olduğuna inanıyor. Kullanıcı büyük olasılıkla kimlik bilgilerinin tehlikeye atıldığının farkında olmaya devam ediyor.

Giriş sayfalarına ek olarak, bazı siteler yönlendirme sayfalarını veya bitiş noktalarını sağlar. Uygulamanızın açık yönlendirmeli bir sayfası olduğunu hayal edin `/Home/Redirect`. Saldırgan, örneğin, giden bir e-postada bir bağlantı oluşturabilir `[yoursite]/Home/Redirect?url=http://phishingsite.com/Home/Login`. Tipik bir kullanıcı URL'ye bakacak ve sitenin adınızla başladığını görecektir. Buna güvenerek, bağlantıyı tıklarlar. Açık yönlendirme daha sonra kullanıcıyı sizinkilerle aynı görünen kimlik avı sitesine gönderir ve kullanıcı muhtemelen siteniz olduğuna inandıklarına giriş yapar.

Açık yönlendirme saldırılarına karşı koruma

Web uygulamaları geliştirirken, kullanıcı tarafından sağlanan tüm verileri güvenilmez olarak değerlendirin. Uygulamanız, kullanıcıyı URL'nin içeriğine göre yönlendiren bir işlevselliğe sahipse, bu tür yönlendirmelerin yalnızca uygulamanızın içinde yerel olarak yapıldığından emin olun (veya sorguda sağlanan herhangi bir URL'yi değil, bilinen bir URL'ye).

LocalRedirect

`LocalRedirect`Temel `Controller`sınıftan yardımcı yöntemi kullanın :

```
C #kopya
public IActionResult SomeAction(string redirectUrl)
{
    return LocalRedirect(redirectUrl);
}
```

`LocalRedirect`yerel olmayan bir URL belirtilirse istisna atar. Aksi takdirde, `Redirect`yöntem gibi davranır .

IsLocalUrl

Yönlendirmeden önce URL'leri test etmek için IsLocalUrl yöntemini kullanın :
Aşağıdaki örnek, yeniden yönlendirmeden önce bir URL'nin yerel olup olmadığını kontrol etmeyi gösterir.

```
C #kopya
private IActionResult RedirectToLocal(string returnUrl)
{
    if (Url.IsLocalUrl(returnUrl))
    {
        return Redirect(returnUrl);
    }
    else
    {
        return RedirectToAction(nameof(HomeController.Index), "Home");
    }
}
```

Bu `IsLocalUrl`yöntem, kullanıcıları yanlışlıkla kötü amaçlı bir siteye yönlendirilmekten korur. Yerel bir URL beklediğiniz bir durumda yerel olmayan bir URL sağlandığında sağlanan URL'nin ayrıntılarını günlüğe kaydedebilirsiniz. Yönlendirme URL'lerinin günlüğe kaydedilmesi, yeniden yönlendirme saldırılarının teşhisine yardımcı olabilir.

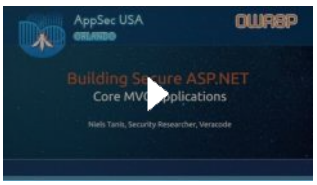
CSRF/XSRF(Cross-Site Request Forgery) Saldırıları Nedir?

Siteler arası istek sahteciliği (ayrıca XSRF veya CSRF olarak da bilinir, belirgin *gör-gezinme*), kötü amaçlı bir web uygulamasının bir istemci tarayıcısıyla bu tarayıcıya güvenen bir web uygulaması arasındaki etkileşimi etkileyebileceği web barındırma uygulamalarına yönelik bir saldırıdır. Bu saldırılar mümkündür, çünkü web tarayıcıları bir web sitesine yapılan her istekle birlikte otomatik olarak bazı türlerde kimlik doğrulama belirteçleri gönderir. Bu istismar biçimi, *tek tıklamayla yapılan bir saldırı* veya *oturum sürme* olarak da bilinir, çünkü saldırı daha önce kimliği doğrulanmış oturumdan yararlanır.
Bir CSRF saldırısı örneği:

1. Bir kullanıcı `www.good-banking-site.com`form kimlik doğrulamasını kullanarak oturum açar. Sunucu kullanıcının kimliğini doğrular ve bir kimlik doğrulama çerezi içeren bir yanıt verir. Site saldırıya açıktır, çünkü geçerli bir kimlik doğrulama çerezi ile aldığı herhangi bir isteği güvenir.
2. Kullanıcı kötü amaçlı bir siteyi ziyaret ediyor `www.bad-crook-site.com`.
   Kötü amaçlı site, `www.bad-crook-site.com`aşağıdakine benzer bir HTML formu içeriyor:
   HTMLkopya
   ```
   <h1>Congratulations! You're a Winner!</h1>
   <form action="http://good-banking-site.com/api/account" method="post">
       <input type="hidden" name="Transaction" value="withdraw">
       <input type="hidden" name="Amount" value="1000000">
       <input type="submit" value="Click to collect your prize!">
   </form>
   ```

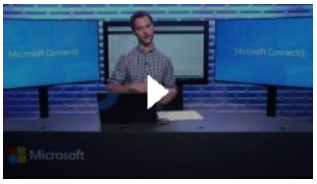
   Formun `action`, kötü amaçlı siteye değil, savunmasız siteye gönderdiğine dikkat edin . Bu, CSRF'nin "siteler arası" kısmıdır.
3. Kullanıcı gönder düğmesini seçer. Tarayıcı isteği yapar ve istenen etki alanı için kimlik doğrulama çerezini otomatik olarak içerir `www.good-banking-site.com`.
4. İstek `www.good-banking-site.com`, kullanıcının kimlik doğrulama içeriğiyle sunucuda çalışır ve kimliği doğrulanmış bir kullanıcının gerçekleştirmesine izin verilen herhangi bir işlemi gerçekleştirebilir.

Building Secure ASP.NET Core MVC Applications - AppSecUSA 2017

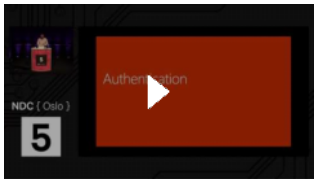


ASP.NET Core 2.0 Security

Securing your web applications with ASP.NET Core 2.0 | E115



Security in ASP.NET Core 2.0 - Barry Dorrans



Building Secure Web APIs with ASP.NET Core