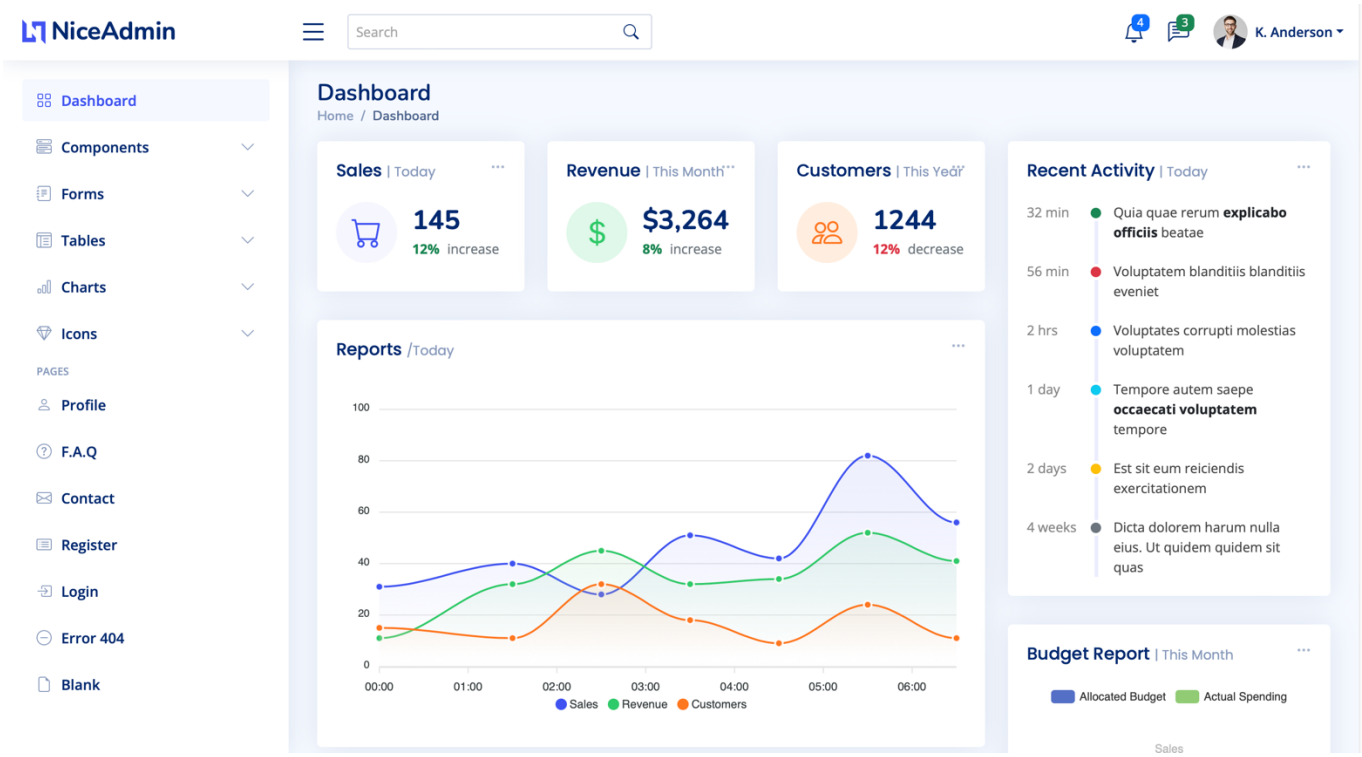
# DASHBOARD TEMPLATE

Projemizde direkt olarak UTM dashboardlarının ücretsiz templateleri bulunmadığından dolayı ücretsiz BootStrap template'i olan NiceAdmin'i kullanacağız. Bu HTML template'i Asp .Net'e projesi oluşturup, entegre edip daha sonrasında üzerinde yalnızca bizim projemizde kullanacağımız bölümlerin kalacağı şekilde kırpıp yol alacağız. Projemiz kapsamında tek sayfa tasarımı olarak bir tasarım düşünüyoruz, detaylar için ekstra sayfalara ihtiyaç olması halinde onları da ekleyeceğiz.

NiceAdmin Demo → https://bootstrapmade.com/demo/NiceAdmin/



**Projemizde güvenli yazılım geliştirme kurallarından gerçekleştireceğimiz ve gerçekleştirebileceğimiz noktalar için checklist;**

✓ Her bir prosedür, modül kendi işini bitirmesi için gerekli en az haklarla çalıştırılmalı ve bu süreç esnasında bu haklara gereken en az süre boyunca sahip olmalı

✓ Her bir nesneye yapılan erişimlerde yetki kontrolü yapılmalı ve bu kontrol normal durumların dışında başlatma, kapatma veya istek, yanıt aşamalarında da yapılmalı

✓ Yüksek güvenlik gerektiren işlemlerde tek bir koşula bağlı olarak izin verilmemeli

✓ Yazılımda kullanılan tüm varsayılan değerler güvenliği arttıracak şekilde seçilmeli ve her bir nesnenin varsayılan erişimi hiç olmalı

✓ Sistem ve yazılım içerisindeki modüller arasında yalıtım sağlanmalı

✓ Yazılımın güvenliği o yazılımdaki en zayıf halkanın güvenliği kadardır. Bu halka tespit edilerek yeniden tasarlanıp gerçeklenmeli

✓ Gereksiz özellikler eklenmemelidir

✓ Yetkisiz bir kullanıcının yazılım ortamından veri alabileceği, girebileceği veya yetkisiz işlemlerde bulunabileceği noktalar tespit edilmeli ve bu noktalar izlenmeli, sınırlandırılmalıdır

? Savunma mekanizmaları peş peşe uygulanmalıdır.

✓ Güvenlik mekanizmaları mümkün olduğunda basit olmalıdır ki bu şekilde hata oluşturduğunda yetkililer tarafından kolay anlaşılmalı ve düzeltilebilmeli

✓ Kullanıcılar işini en kolay şekilde ancak en az yetkiyle yapabilmeli

? Uygulamanın güvenlik katmanları birbirinden ayrılmalı ve güvenlik işlevlerinin normal işlemlerin gerçekleştirildiği çalışma ortamından farklı bir izole ortam içerisinde gerçekleştirilmesi sağlanmalıdır.

✓ Uygulama veritabanında kişisel veri içeren birincil anahtar (kimlik no, e-posta adresi vb.) kullanılmamalıdır.

✓ Sistem mimarisi zayıf yönleri veya zayıf noktaları bulmak için saldırganın bakış açısı ile incelenmeli.

✓ Kaynak kodun zafiyet ve faaliyet analizleri yapılmalı.

✓ Güvenli diller, güvenli ve onaylanmış kütüphaneler kullanılmalıdır. Eski ve güvensiz kütüphaneler kullanılmamalıdır.

✓ Kasıtlı veya kasıtsız uygulamada güvenlik açığı yatacak kodu kimin yarattığının kontrolü için sürüm kontrolü yapılmalıdır.

✓ Kod kalitesi ölçümünde pep8 vb. kodlama stilleri üzerinden kontroller yapılır. Bu kontroller kodlamanın yazım biçimlerini, kod içinde bulunan boşluklar, değişken isimleri, fonksiyonların girdi sayılarını, satır sayılarını ve benzeri özellikleri kontrol eder.

✓ Otomatik araçlar ürettiği hata çıktılarının bir kısmı yanlış alarm (false pozitif) hatalardır. Bundan dolayı otomatik araçların çıktıları manuel olarak da incelenmelidir.

? Koruma gerektiren kritik verinin tam olarak korunabilmesi için, aynı zamanda uygulama kaynak kodu, binary kodu ve çalışma zamanı kodunun da korunması gerekmektedir.

✓ Mimaridekı tüm yazılım bileşenleri tanımlı olmalı ve ihtiyaç duyulmayan bileşenler kaldırılmalıdır.

? Uygulama saldırıya uğradığında izlenecek saldırı karşılık planı oluşturulmalıdır.

? Periyodik olarak veritabanları, uygulamalar ve uygulama verisi yedeklemesi yapılmalıdır.

✓ Gereksinimler açık, tutarlı, tam, uygulanabilir, takip edilebilir ve doğrulanabilir olmalıdırlar.

✓ Tüm parola alanlarında kullanıcı giriş yaparken kullanıcının parolası maskelenmeli ve açık olarak görünmemelidir.

✓ Kimlik doğrulama başarısız olduğu takdirde güvenli bir duruma geçilmeli ve saldırganların yetkisiz oturum açmaları engellenmelidir.

✓ Hesaba yeniden erişebilecek tüm hesap kimlik doğrulama işlevleri (profil güncelleme, parolamı unuttum vb.) en az ana kimlik doğrulama mekanizması kadar saldırılara dayanıklı olmalıdır.

? Hassas işlevler gerçekleştirilmeden önce, yeniden kimlik doğrulama, daha güçlü bir mekanizmayla kimlik doğrulama, ikinci faktör veya işlem imzalama gibi yöntemler uygulanmalıdır.

✓ Kaynak kodunda veya kaynak kodu depolarında gizli bilgiler, API anahtarları ve parolalar mevcut olmamalıdır.

✓ Uygulamanın yönetim arayüzlerine güvenilmeyen taraflarca erişilmesi engellenmelidir.

✓ Uygulama kullanıcının son başarılı oturum açma tarih ve saatini görüntülemelidir.

✓ Güvenilmeyen kaynaklardan alınan dosyaların türü doğrulanmalı ve zararlı bir içeriğe sahip olup olmadığı kontrol edilmelidir.

✓ Oturumlar belirli bir süre etkinlik olmadığında kendiliğinden sonlanmalıdır.

✓ Kimlik doğrulamayla erişilen tüm sayfalardan oturum kapatma işlevine erişilebilmelidir.

✓ Oturum kimliğinin URL, hata mesajları ve iz kayıtları içerisinde yer almaması sağlanmalıdır. URL içerisinde oturum kimliğinin yeniden yazılması engellenmelidir.

✓ Tüm kimlik doğrulama ve yeniden kimlik doğrulama işlemleri sonucunda yeni bir oturum ve yeni bir oturum kimliği üretilmelidir.

✓ Kullanıcı sadece yetkilendirildiği uygulama bileşenlerine ve kaynaklara erişebilmeli ve bunları kullanabilmelidir.

✓ Bellekte tutulan önemli veriler gereksinimi sona erdiğinde güvenlik ihlali oluşturamayacak şekilde silinmelidir.

✓ Uygulama, hassas veri ve kişisel verileri içeren hata mesajı veya iz kaydı üretmemelidir.

✓ Uygulama tarafından, istemci ve sunucu tarafında, kabul edilen her bir veri tipi için girdi doğrulama denetimi yapılmalıdır.

✓ HTML form alanlarının veri girdileri, REST çağrıları, HTTP üst başlıkları, çerezler, toplu işlem dosyaları, RSS beslemeleri gibi veri girdileri için doğrulama denetimi yapılmalıdır.

✓ Uygulama, sunucu ve istemci tarafında dil kodlaması (encoding) saldırılarına karşı dayanıklı olmalıdır.

✓ Uygulama, kişisel verileri şifreli olarak saklamalı ve bu verilerin taşınmasında korumalı iletişim kanallarını kullanmalıdır.

✓ Uygulamanın istemci tarafında çalışan kodları, kişisel verileri başka ortamlara aktarmamalı (konsola yazma, başka dosya olarak kaydetme, yerel veya uzak uygulamalara transfer etme vb.), güvensiz ortamlarda (ortak dizin, USB disk vb.) güvensiz yöntemlerle (açık metin olarak, zayıf şifreleme algoritma kullanarak şifreleme vb.) saklamamalıdır.

✓ Kullanılan veritabanının dışarıya aktarımı ancak veritabanı yönetim yetkisi olan hesaplarla yapılmalı ve öncesinde veritabanındaki kişisel verilerin silinmesi sağlanmalıdır.

✓ Değişen parola fonsiyonu eski parolayı, yeni parolayı ve bir parola onayını kapsamalıdır.

✓ Kaba kuvvet saldırıları ya da servis dışı bırakma saldırıları gibi otomatik yapılan yaygın kimlik doğrulama saldırılarını önlemek için istekler azaltılmalıdır. Aşırı kimlik doğrulama denemelerini engellenmeli.

✓ Parolalar için bir en uzun geçerlilik süresi tanımlanmış olmalıdır.

✓ Uygulama, ayar ve denetim dosyaları kullanıcı verisiyle aynı konumda depolamamalıdır.

✓ Desteklenmeyen, güvensiz veya teknolojisi zaman aşımına uğramış istemci teknolojileri kullanılmamalıdır.

? Uygulama tarafından etkin ve aynı zamanlı oturumların sayısı sınırlandırılabilmelidir. Sızma testi ile test edilebilir.

? Her bir kullanıcının uygulamadaki etkin oturumları görüntülenebilmeli, kullanıcı herhangi bir etkin oturumunu sonlandırabilmelidir.

✓ Web uygulamalarında oturum çerezlerinde HTTPOnly bayrağı etkin olmalıdır.

? Uygulama, başarısız sistem başlatma, başarısız sonlandırma veya başarısız kapatma gibi işlemlerde güvenli bir duruma geçmelidir. Negatif test ile test edilebilir.

? Tüm kriptografik modüllerin, güvenli bir şekilde hataya düştüğü doğrulanmalıdır. Hata yönetimi “Oracle Padding” atağına imkan tanımayacak şekilde olmalıdır. Negatif test ile test edilebilir.

✓ Tüm rastgele üretilen sayılar, dosya isimleri, global eşsiz değerler (GUID) ve karakter dizilerinin saldırgan için tahmin edilemez olması sağlanmalıdır. Rastgele sayıların yüksek entropiye sahip olarak üretilmelidir. Negatif test ile test edilebilir.

✓ Tüm anahtar ve şifreler kullanımları tamamlandığında, tamamen sıfırlanarak yok edilmelidir. Negatif test ile test edilebilir.

✓ Tüm formlarda istemci tarafında yapılan ön bellekleme işlevselliği önemli veriler için kapatılmalıdır.

✓ Siteler arası komut dosyası çalıştırma (XSS) engellemede uygulama-istemci arasındaki hassas veri iletişim için HTTP başlığı veya gövdesi kullanılmalıdır. Postman, Wireshark vs. ile test edilebilir.

✓ Uygulama, saklama gereksinimi sona erdikten sonra önemli verileri güvenlik sonunu yaratmayacak şekilde silinmelidir.

? Sunucuya gelen isteklerin öngörülmeyen bir sayıda ya da büyüklükte olup olmadığı kontrol edilebilmelidir. Negatif test ile test edilebilir.

✓ İz kayıtlarında olayların zaman sıralamasına ilişkin araştırma yapılabilecek şekilde zaman bilgisi yer almalıdır.

✓ Uygulama tarafından üretilen iz kayıtları hassas bilgi içermemelidir.

✓ Uygulama hassas bilgileri formlarda bulunan gizli alanlarda saklamamalıdır.

✓ Uygulama Cross-Site Request Forgery (CSRF)'dan kaynaklanan açıklıklardan korunma mekanizmasına sahip olmalıdır. Fuzz testi, kaynak kod zayıflık analizi ile test edilebilir.

? Uygulamanın çalışma ortamı, bellek taşması saldırılarına dayanıklı olmalıdır veya mevcut güvenlik mekanizmaları bellek taşmasını engellemelidir.

✓ Sunucuda yapılan girdi doğrulama hataları, isteğin reddi ile sonuçlanmalı ve iz kaydı oluşturulmalıdır.

✓ SQL Injection engellemek için bütün veritabanı sorguları, parametre olarak yapılmalı ve veritabanına erişimde kullanılan dile karşı önleyecek denetimler yapılmalıdır. Veritabanı açıklık tarama aracı, web uygulama zayıflık tarayıcı veya negatif test ile test edilebilir.

? LDAP-Active Directory Injection için uygulama, yetki onaylama hizmetlerinin enjeksiyonu açıklıklarını önleyici güvenlik denetimlerini yapmalıdır. Tasarım gözden geçirme, işlevsel test, negatif test ve kaynak kod gözden geçirme ile test edilebilir.

✓ İşletim sistemi komut enjeksiyonu için uygulama, işletim sistemi komut enjeksiyonu açıklıklarını önleyici güvenlik denetimlerini yapmalıdır. Kaynak kodu açıklık tarayıcısı, Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test, kaynak kod gözden geçirme ile test edilebilir.

✓ Uygulama, girdi alınan içerik bir dosya yolu içeriyorsa, uzak ya da yakın dosya içerme açıklıklarını önleyici güvenlik denetimlerini yapmalıdır.

✓ Uygulama, XML açıklıklarını (XPath sorgu saldırıları, XML harici öğe saldırıları, XML enjeksiyonu vb.) önleyici güvenlik denetimlerini yapmalıdır.

✓ Uygulama, web servislerini iyi yapılandırılmış en az TLS v1.2 veya SSL gibi muadil güvenlik önlemi sunan bir protokol ile sunacak şekilde tasarlanmalıdır.

✓ Uygulama, web servis kimlik doğrulama ve yetkilendirmesi için oturum temelli yapılar kullanacak şekilde tasarlanmalıdır.

✓ Uygulama, web servisi ile gönderilen veride betik (script) içermeyecek şekilde tasarlanmalıdır. Statik analiz ile test edilebilir.

✓ Uygulama, web servislerinden şifreli olarak paylaşılan verileri yine şifreli olarak saklayacak şekilde tasarlanmalıdır.

✓ Uygulama, kişisel veriler üzerinde işlem yapılması ana amaç olmayan durumlarda kişisel verileri maskeleyerek görüntülemeli, aktarmalı veya işlemelidir.