WEB PENTEST RAPOR

HAZIRLAYAN

Alaeddin AR

Eylül 2022

GİRİŞ

Bu rapor, Alaeddin AR tarafından php.testspaerker.com sitesi üzerindeki güvenlik açıklarını ortaya çıkartmak amacı ile 21.08.2022-10.09.2022 tarihleri arasında gerçekleştirilen güvenlik ve sızma testlerinin (penetration test) detaylı sonuçlarını içermektedir**.**

KAPSAM

Sızma testinde ana amaçlardan biri tüm zafiyetlerin değerlendirilerek sisteme sızılmaya çalışılmasıdır. Bu amaç doğrultusunda gerçekleştirilecek sızma testlerinde kapsam pentest çalışmasının en önemli adımını oluşturmaktadır.

Web Uygulama Güvenliği: Uygulama seviyesi açıklıklar genel olarak kullanılan programlama dilindeki kontrol eksikliği ve son kullanıcıdan alınan girdilerin yeterli kontrolden geçirilmemesinden kaynaklanmaktadır.

Yansıtılan Siteler Arası Script Çalıştırma (XSS) -1

Bulgu Açıklaması:

Kalıcı olmayan XSS olarak da bilinen reflected XSS, bilgisayar korsanları kötü amaçlı komut dosyasını doğrudan bir HTTP isteğine enjekte eder. Ardından, web sunucusundan yürütüldüğü kullanıcının tarayıcısına yansıtır. Bilgisayar korsanı sıklıkla hedeflenen kişilere, onları savunmasız bir sayfaya getiren özelleştirilmiş bağlantılar gönderir.

Reflected XSS saldırıları kalıcı değildir. Bir kullanıcı kötü niyetli bir bağlantıyı tıkladığında, özel olarak hazırlanmış bir formun göndermesi veya kötü niyetli bir siteye göz atması için kandırıldığında, enjekte edilen kod savunmasız web sitesine gider. Web sunucusu, sırayla, enjekte edilen komut dosyasını kullanıcının tarayıcısına döndürür veya yansıtır. Bu aldatma, bir hata mesajında, arama sonucunda veya isteğin bir parçası olarak sunucuya gönderilen verileri içeren başka bir yanıt türünde olabilir. Tarayıcı, yanıtın, kullanıcının zaten etkileşimde bulunduğu "güvenilir" bir sunucudan geldiğini varsaydığı için kodu yürütür.

URL: http://php.testsparker.com/artist.php?id=

Http Talep Türü: GET

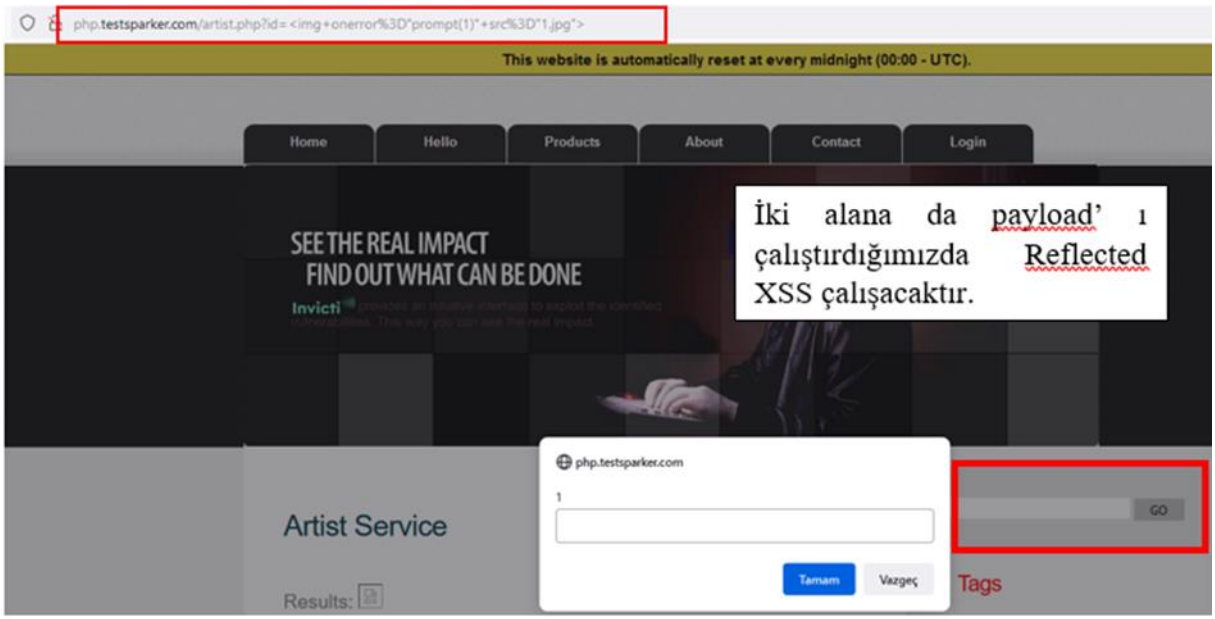
Payload: <img onerror="prompt(1)" src="1.jpg">

Parametre: name

Hedefe Gönderilen GET isteği:

http://php.testsparker.com/artist.php?id=%3Cimg+onerror%3D%22prompt%281%29%22+src%3D%221.jpg%22%3E

Bu verilen bilgiler doğrultusunda web sayfasında url kısmına payload ' ekler veya arama kısmına belirtilen payload çalıştırıldığı zaman Reflected XSS çalışacaktır.



Açıklığı Barındıran Sistemler:

http://php.testsparker.com/artist.php?id=

http://php.testsparker.com/auth/internal.php

Çözüm Önerileri:

Uygulama kodlarının gözden geçirilerek parametreler ve http başlığındaki diğer alanlar vasıtası ile yollanan her türlü bilginin kullanılmadan önce zararlı karakterlerden filtrelenmesi önerilmektedir. Uygulamalardaki bütün girdi ve çıktı noktalarından gelen değişkenler kontrole tabi tutulmalı ve bu girdilerdeki bütün meta karakterler filtrelenmelidir. Detaylı XSS önleme yöntemleri için aşağıda belirtilen kaynaklar incelenebilir.

https://owasp.org/www-community/attacks/xss/

https://www.cgisecurity.com/xss-faq.html

Depolanan Siteler Arası Script Çalıştırma (XSS)-2

Bulgu Açıklaması

Bilgisayar korsanları yüklerini güvenliği ihlal edilmiş bir sunucuda depoladığında saldırılar gerçekleşir. Genellikle zarar veren bir XSS saldırı yöntemidir. Saldırgan, yüklerini hedef uygulamaya enjekte etmek için bu yaklaşımı kullanır. Uygulamanın giriş doğrulaması yoksa, kötü amaçlı kod, uygulama tarafından veri tabanı gibi bir konumda kalıcı olarak depolanır veya kalıcı olur. Pratikte bu, saldırganın bir blog veya forum gönderisindeki yorum bölümleri gibi kullanıcı giriş alanlarına kötü amaçlı bir komut dosyası girmesine olanak tanır.

Saldırganın yükü, virüslü sayfayı açtığında, tarayıcısında meşru bir yorumun görünmesiyle aynı şekilde, kullanıcının tarayıcısına sunulur. Hedeflenen kişiler, sayfayı tarayıcılarında görüntülediklerinde yanlışlıkla kötü amaçlı komut dosyasını yürütürler.

Bulgu 1:

URL: http://php.testsparker.com/artist.php?id=

Http Talep Türü: GET

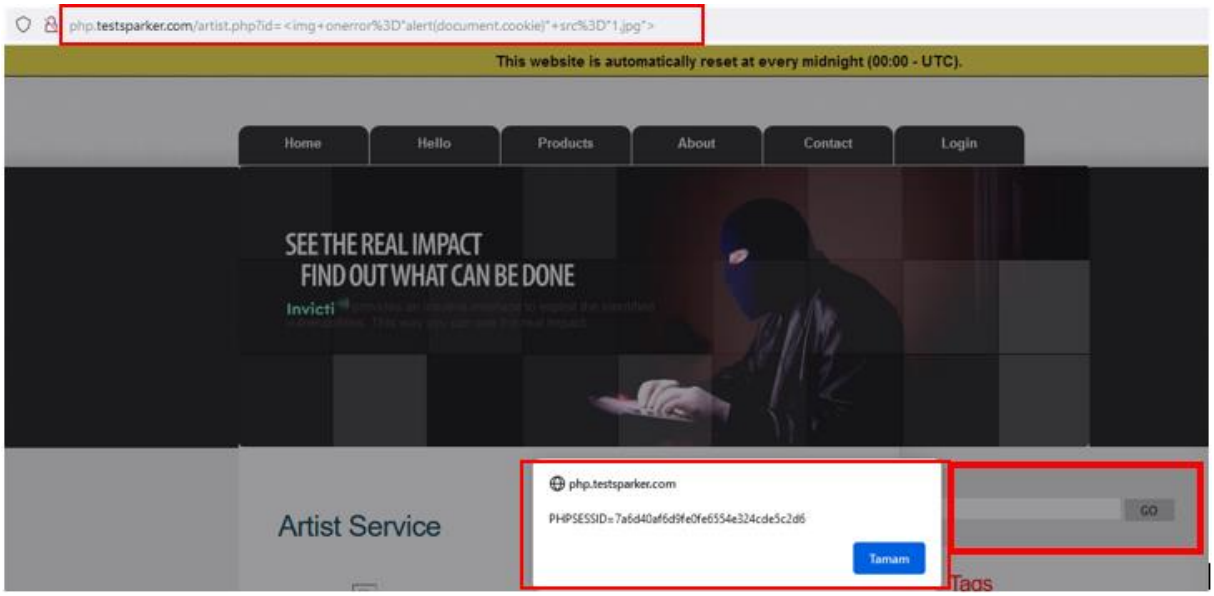
Payload: <img onerror="alert(document.cookie)" src="1.jpg">

Parametre: name

Hedefe Gönderilen GET isteği:

http://php.testsparker.com/artist.php?id=%3Cimg+onerror%3D%22alert%28document.cookie%29%22+src%3D%221.jpg%22%3E

Bu verilen bilgiler doğrultusunda web sayfasında url kısmına payload ‘ ekler veya arama kısmına belirtilen payload çalıştırıldığı zaman Store(Depolanan) XSS çalışacaktır.



Bulgu 2:

URL: http://php.testsparker.com/artist.php?id=

Http Talep Türü: GET

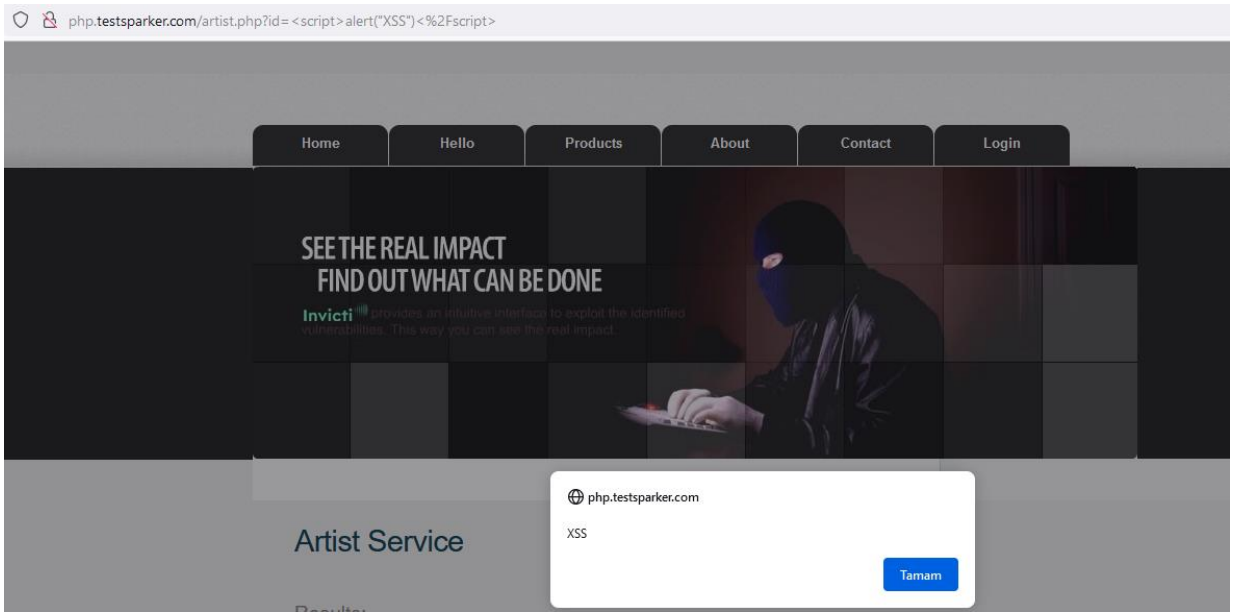
Payload: <script>alert("XSS")</script>

Parametre:

Hedefe Gönderilen GET isteği:

http://php.testsparker.com/artist.php?id=%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E

Bu verilen bilgiler doğrultusunda web sayfasında url kısmına payload ‘ ekler veya arama kısmına belirtilen payload çalıştırıldığı zaman Store(Depolanan) XSS çalışacaktır



Bulgu 3:

URL: http://php.testsparker.com/artist.php?id=

Http Talep Türü: GET

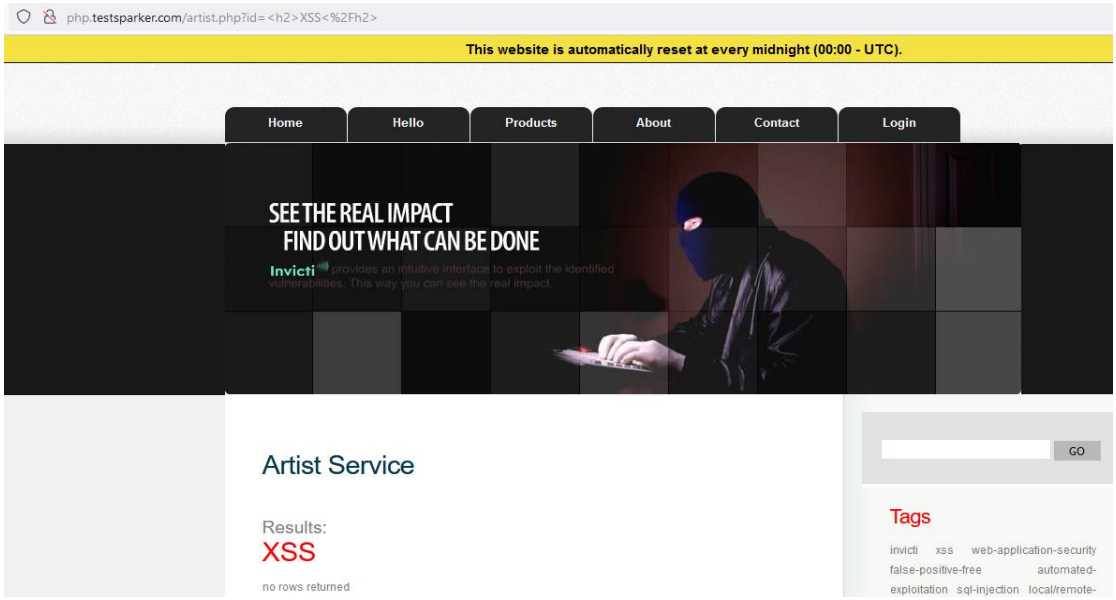
Payload: <h2>XSS</h2>

Parametre: name

Hedefe Gönderilen GET isteği:

http://php.testsparker.com/artist.php?id=%3Ch2%3EXSS%3C%2Fh2%3E

Bu verilen bilgiler doğrultusunda web sayfasında url kısmına payload ‘ ekler veya arama kısmına belirtilen payload çalıştırıldığı zaman Store(Depolanan) XSS çalışacaktır.



Açığı Barındıran Sistemler:

http://php.testsparker.com/artist.php?id=

http://php.testsparker.com/auth/internal.php

Çözüm Önerileri:

Uygulama kodlarının gözden geçirilerek parametreler ve http başlığındaki diğer alanlar vasıtası ile yollanan her türlü bilginin kullanılmadan önce zararlı karakterlerden filtrelenmesi önerilmektedir. Uygulamalardaki bütün girdi ve çıktı noktalarından gelen değişkenler kontrole tabi tutulmalı ve bu girdilerdeki bütün meta karakterler filtrelenmelidir. Detaylı XSS önleme yöntemleri için aşağıda belirtilen kaynaklar incelenebilir.

https://owasp.org/www-community/attacks/xss/

https://www.cgisecurity.com/xss-faq.html

SQL Injection Zafiyeti-3

Bulgu Açıklaması

SQL Injection zafiyeti, uygulama parametreleri aracılığı ile yollanan bilgilerin düzgün kontrol edilmemesi sebebi ile arka planda çalışan veritabanına yollanan sorgulara, saldırganın sorgularını eklemesine imkan tanıyan bir güvenlik açığıdır.

Hata Tabanlı SQL Injection saldırıları, uygulamanın veri tabanına gönderdiği sorgularda herhangi bir yazım hatası syntax error olması durumunda veya sorgunun veri tabanında çalışması sonucu dönen verilerin, ekrana çıktı olarak verilmesi temeline dayanır.

URL: http://php.testsparker.com/artist.php?id=

Http Talep Türü: GET

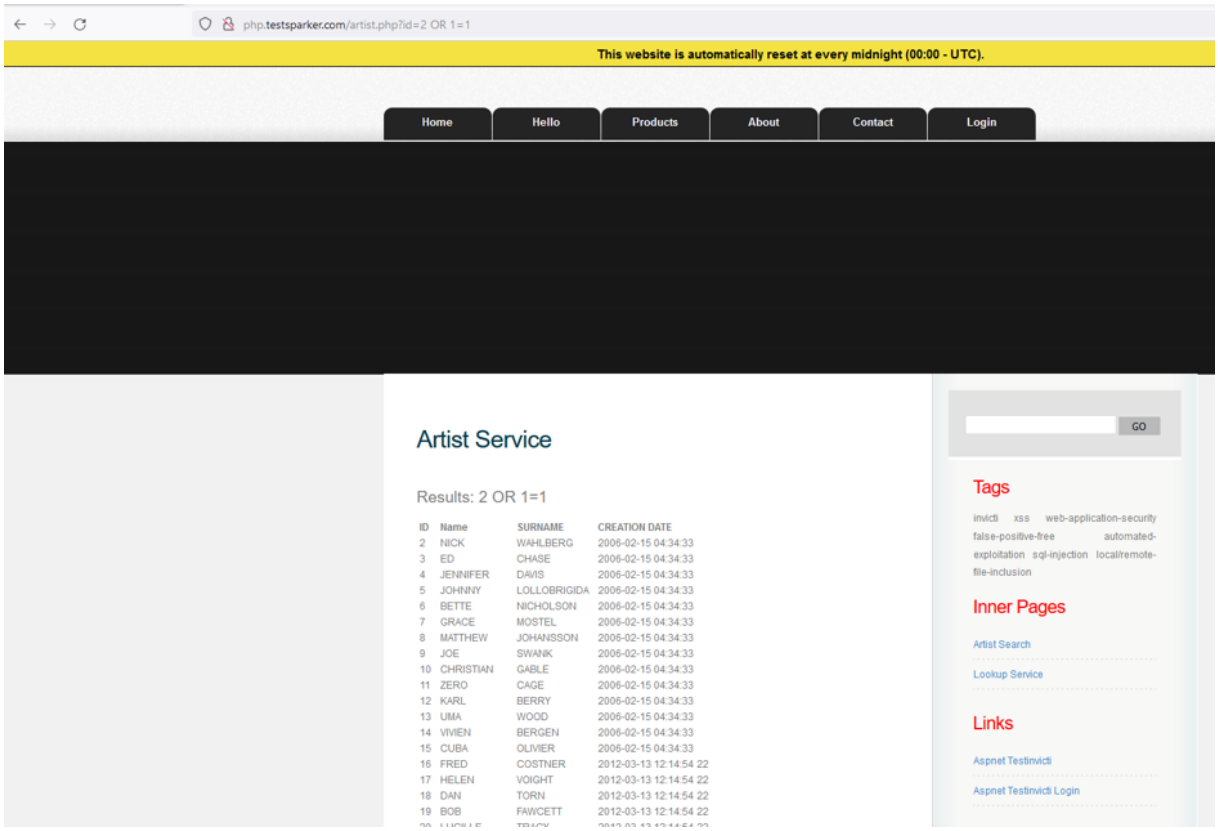
Payload: 2 OR 1=1

Parametre:

Hedefe Gönderilen GET isteği:

http://php.testsparker.com/artist.php?id=2%20OR%201=1

Bu verilen bilgiler doğrultusunda web sayfasında url kısmına payload ‘ eklediğimiz zaman veri tabanı içerisinde kullanıcılara ulaşılabilir.



Açığı Barındıran Sistemler:

http://php.testsparker.com/artist.php?id=

Çözüm Önerileri:

Uygulama kodlarının gözden geçirilerek parametreler ve http başlığındaki diğer alanlar vasıtası ile yollanan her türlü bilginin kullanılmadan önce zararlı karakterlerden filtrelenmesi önerilmektedir. Uygulamalardaki bütün girdi noktalarından gelen değişkenler girdi kontrolüne sokulmalı ve bu girdilerdeki bütün meta karakterlerin filtrelenmesi önerilmektedir. Detaylı SQL enjeksiyonu önleme yöntemleri için aşağıda belirtilen kaynak incelenebilir.

https://owasp.org/www-community/Injection_Flaws

Yetersiz Kimlik Doğrulama (Broken Authentication)-4

OWASP, yetersiz kimlik doğrulama hatalarından faydalanarak gerçekleştirilebilen saldırıları 3 ana başlık altında açıklamaktadır:

- Kullanıcı bilgilerini deneme (Credential Stuffing)
- Kaba kuvvet saldırıları ile erişim (Brute Force Access)
- Oturum çalma (Session Hijacking)

Kullanıcı bilgilerini deneme (Credential Stuffing) çeşitli veri ihlalleri sonucu internete düşmüş olan kullanıcı adı ve parola kombinasyonlarından oluşturulan listeleri kullanarak otomatize araçlar ile kullanıcı girişi yapmaya çalışmaktır. Kullanıcıların genellikle farklı platformlarda aynı parolayı kullanma alışkanlığı olduğundan saldırganlar bu yöntem ile zaman zaman başarı sağlayabilmektedir.

Kaba kuvvet saldırıları ile erişim yöntemi, her bir parola ihtimalinin denenmesi ile doğru parolanın bulunmasına çalışılması anlamına gelmektedir. Saldırganlar internette kolay bir arama ile erişilebilen "sık kullanılan parolalar" listelerinden faydalanarak bu parolaları deneyen betikler kullanır ve otomatik araçların kullanılan parolayı bulmasını sağlamaya çalışır.

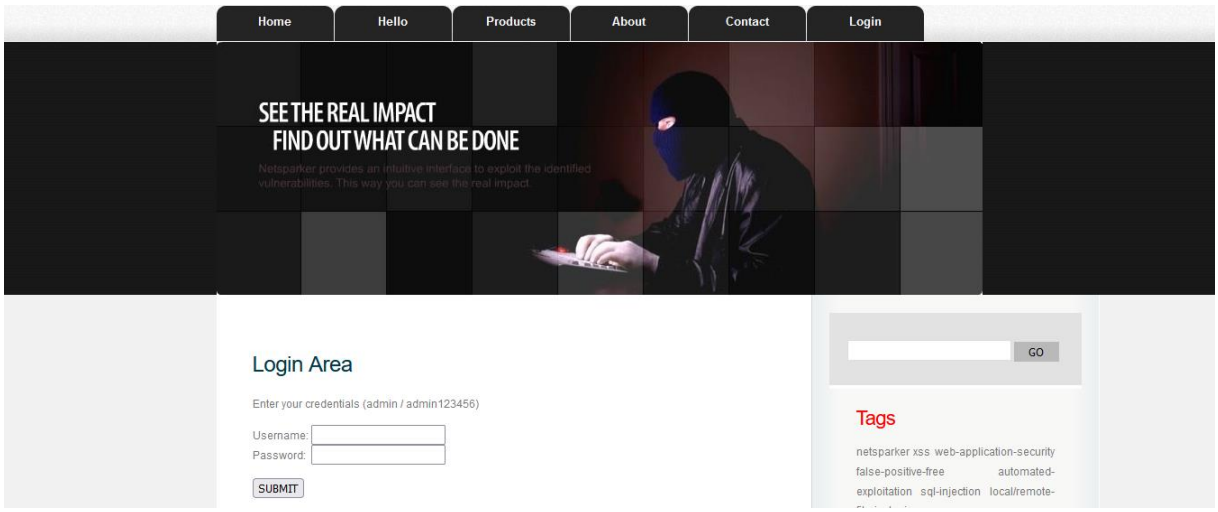
Oturum çalma meşru bir kullanıcının kimliği doğrulanmış oturumunun kötüye kullanılmasıdır. Oturum açıldıktan sonra, sistem genellikle kullanıcıya bir oturum kimliği (session ID) atar, böylece ziyaret edilen her yeni sayfa için yeniden oturum açmaya gerek kalmaz. Bu oturum kimliği, genellikle tarayıcıdaki URL'ye eklenen bir sayı veya kullanıcının cihazına yerleştirilen bir oturum çerezidir. Teorik olarak, kullanıcı oturumu kapattığında oturumdan kaldırılır. Saldırganlar, trafiği izleyerek oturum kimliğini elde edebilirse, meşru kullanıcının oturumunu ele geçirebilir. Mevcut kullanıcının kimliği zaten doğrulanmış olduğundan, saldırgan o kullanıcıya izin verilen herhangi bir eylemi gerçekleştirebilir.

URL: http://php.testsparker.com/auth/internal.php

K.adı: admin

Parola: admin123456

Verilen URL belirtilen kullanıcı adı ve parola ile giriş yapınca giriş yapabiliyoruz.



Açığı Barındıran Sistemler:

http://php.testsparker.com/auth/internal.php

Çözüm Önerileri:

NIST tarafından yayınlanan önerilerde aşağıdaki parola seçimlerinin yasaklanması gerektiği belirtilmiştir:

- Daha önceki veri ihlallerinde çalındığı bilinen parolalar ("En sık kullanılan parolalar" şeklinde yapılacak bir arama sonucunda çıkan parolaların kullanımı engellenebilir).
- Tek kelime içeren parolalar (sözlükte bulunabilen kelimeler).
- Aynı karakterin tekrar ettiği parolalar (aaaaaa, 123456, 1234abcd vb.)
- Kullanıcı adının, soyadının, doğum tarihinin parola olarak kullanımı
- Kullanılan servisin adının parola olarak belirlenmesi

Yetersiz Kimlik Doğrulama için aşağıda belirtilen kaynaklar incelenebilir.

https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication