Açık Kaynak Kodlu Web Uygulaması Güvenlik Duvarı (WAF) Güvenlik Testleri

ÖZET

İletişim teknolojilerinin hızla gelişmesi ve internetin yaygınlaşması, günlük yaşamımızda web sitelerinin önemini büyük ölçüde artırmıştır. Web siteleri, eğitimden ticarete, eğlenceden haberleşmeye kadar pek çok alanda hayatımızı kolaylaştırmakta ve bu nedenle güvenlikleri kritik bir hale gelmektedir. Ancak, hızlı ve sonuç odaklı geliştirilen uygulamalar, çeşitli zafiyetler barındırmakta ve kötü niyetli kullanıcılar için fırsatlar yaratmaktadır. Bu durum, web sitelerinin güvenliğini sağlama ihtiyacını her geçen gün daha da artırmaktadır. Özellikle e-ticaret gibi hassas bilgiler içeren uygulamalarda güvenliğin sağlanması hayati önem taşımaktadır.

Web uygulama güvenliğini artırmak için, zafiyetleri tespit etmek ve engellemek gereklidir. Geliştirme aşamasında alınacak önlemlerin yanı sıra, uygulama canlı ortama alındığında da çeşitli güvenlik tedbirleri uygulanmalıdır. Bu tedbirlerden biri de web uygulama güvenlik duvarları (WAF) kullanmaktır. WAF'lar, belirli kurallar çerçevesinde çalışarak saldırıları tespit eder ve engeller. Birçok firma, bu güvenlik duvarlarını kullanarak olası saldırıları önlemeye çalışmaktadır.

Bu proje kapsamında, Linux sunucusu üzerinde çalışan ve çeşitli zafiyetler içeren Damn Vulnerable Web Application (DVWA) yazılımının önüne bir Web Uygulama Güvenlik Duvarı (WAF) kurulumu gerçekleştirilmiştir. WAF olarak, geniş güvenlik özellikleri sunan ModSecurity kullanılmıştır. Proje sürecinde, DVWA'ya yönelik çeşitli saldırı senaryoları oluşturularak WAF'ın etkinliği test edilmiştir. Bu testlerde, SQL enjeksiyonları, XSS (Cross-Site Scripting) saldırıları ve dosya yükleme zafiyetleri gibi yaygın web uygulama güvenlik açıkları kullanılmıştır. Sonuçlar, WAF'ın belirli saldırı türlerine karşı ne kadar etkili olduğunu ve DVWA'nın güvenliğini ne ölçüde artırdığını değerlendirmektedir. Bu çalışmada, gerçekleştirilen testlerin detayları ve WAF'ın performansını değerlendiren bulgular sunulmuştur.

1.GİRİŞ

Saldırganların ağ ve sistem güvenlik açıklarını öğrenip sistemdeki bilgileri ele geçirmeye çalışması, güvenlik duvarı ve saldırıları tespit etmek amacıyla kullanılan sızma testlerine olan talebi artırmaktadır. Siber güvenlik alanında çalışan araştırmacılar, mevcut yazılımları iyi yönetmekte ve güvenlik ürünlerinin çeşitlenmesi ile bilgi güvenliği alanındaki ihlallerin uygulama seviyesine dönüştüğünü gözlemlemektedir. Sızma testlerini başarıyla geçen uygulamalar, sistem yöneticileri tarafından daha fazla tercih edilmekte ve güvenliği sağlamada etkin rol oynamaktadır. Başarısız olan uygulamalar ise kullanıcıların beklediği faydayı sağlayamamakta ve sistemi daha saldırıya açık hale getirmektedir. Bu nedenle web uygulama güvenlik duvarına (WAF) duyulan ihtiyaç giderek artmaktadır.

Web uygulama güvenlik duvarı, istemcilerin sunucuya ulaşmadan önce trafiği durdurarak sadece belirlenen isteklerin sunucuya iletimini sağlar. Web uygulama güvenlik duvarı (WAF), uygulama ile istemci arasına konumlandırılarak iki yönlü trafiği yönetir. Bu çalışmada kullanılan açık kaynaklı güvenlik uygulaması olarak ModSecurity tercih edilmiştir. WAF'ların sağladığı güvenliğin, değişken durumlar üzerinden ne kadar önemli olduğunu göstermek ve bu konudaki eksikliklerin giderilmesi gerektiği vurgulanmıştır.

Literatürde web güvenlik zafiyetlerini incelemek için birçok çalışma bulunmaktadır. Bunların bazıları güvenlik duvarlarını incelerken diğer bir bölümü de spesifik açıkları incelemişlerdir.

Khandelwal ve arkadaşları 2013 yılında yaptıkları çalışmada, web uygulamalarında bulunan güvenlik zafiyetlerini engellemek için ön-cephe engelleme teknikleri hakkında çalışma yapmışlardır. Yaptıkları çalışmada saldırganların nasıl test edildiğini, çok sayıda web uygulama güvenlik zafiyetlerinin tespitinden faydalandığını ve bu zafiyetlerin nasıl engelleneceği hakkında bilgiler vermişlerdir.[1].

Erhan SAYGILI ve Fatih KILIÇ 2019'da yaptıkları araştırmada, çeşitli zafiyetler içeren Ubuntu Server üzerinde çalışan Damn Vulnerable Web Application (DVWA) yazılımına, savunmasız bir sistem ile güvenlik duvarı kullanarak, sızma testleri yapmışlardır ve elde edilen sonuçları kendi çalışmalarında sunmuşlardır.[2].

Murat ALAGÖZ 2020'de yaptığı çalışmada, ModSecurity açık kaynak web uygulama güvenlik duvarının kullanılabilirlik teknikleri ile analizini gerçekleştirmiş, elde edilen bulgular ışığında Mod Security'nin kullanılabilirliğinin arttırılması için gerekli önlemler sunarak alınan önlemlerin kullanıcı çalışması ile doğrulayarak kullanılabilirliğini artıracak öneriler sunmuştur.[3].

Bu çalışmada Linux Server üzerinden içerisinde çeşitli zafiyetleri barındıran Damn Vulnerable Web Application (DVWA) yazılımı önüne web uygulama güvenlik duvarı (WAF) kurularak birçok sızma testi yapılmıştır.WAF'a çeşitli saldırılar yaparak ne kadar başarılı/başarısız olduğu araştırılmıştır.

1.1 WAF NEDİR?

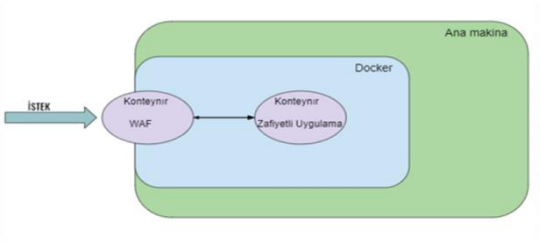
Web Uygulama Güvenlik Duvarı (WAF), web uygulamalarını çeşitli saldırılardan koruyan bir güvenlik sistemidir. WAF, istemci ile sunucu arasına konumlandırılır ve gelen-giden trafiği izleyerek zararlı istekleri engeller.

WAF'lar, belirli kurallar ve politikalar kullanarak güvenliği sağlar ve böylece web uygulamalarının güvenliğini artırarak saldırılara karşı daha dirençli hale getirir.

1.2 WAF NASIL ÇALIŞIR?

WAF'lar yaygın olarak ters proxy olarak çalışır. Bu özelliğe göre gelen tüm trafik WAF üzerinden geçmektedir. Diğer çalışma modları ise şeffaf vekil, köprü, hat dışı ağ izleme ve sunucu tabanlıdır.

Yapılan çalışmada Docker kullanılarak ModSecurity WAF üzerinden incelemeler yapılmıştır. Docker, containerlar içerisinde process'lerin izole olarak çalışmasını sağlayan, uygulamaları hızlı bir şekilde düzenlemeye, test etmeye ve dağıtmaya yarayan bir teknolojidir. Peki WAF'lar Docker kullanılarak nasıl çalışır ?



Şekil 1.2. WAF genel mimarisi

Kullanıcıdan gelen HTTP istekleri, güvenlik duvarlarında tanımlanan kurallara göre analiz edilerek saldırılar önlenmeye çalışılır. Web Uygulama Güvenlik Duvarı (WAF) önce gelen isteği inceler. Eğer gelen HTTP paketi geçerli bir paket ise, bu paket zafiyetli konteynıra iletilir. WAF ve zafiyetli konteynır arasında çift yönlü bir iletişim olduğu için konteynır çıktısını WAF'a verir, WAF da bu çıktıyı kullanıcıya iletir. İkisi de farklı konteynırlar içerisindedir ancak aynı ağ içerisinde yer alırlar. Güvenlik duvarları, zafiyetli web sitelerinin önüne konularak Docker aracılığıyla devreye alınır. Gelen isteklerin önce kendine ulaşmasını sağlar ve sahip olduğu kurallara göre zararlı olarak tespit edilen saldırıları engeller.

1.2 LİTERATÜR TARAMASI:

Wool(2004) tarafından yapılan çalışmada, kurumsal güvenlik duvarlarının genellikle kötü yazılmış kural setlerini uyguladığını belirtmiş, kural seti kalitesini iyileştirmek için bazı yararlı gözlemler sunmuştur.

Raja ve diğerleri (2009) Güvenlik Duvarı Kullanıcılarının Zihinsel Modellerini İyileştirme çalışmasında, tasarımcıların kişisel güvenlik duvarında kullanılan ara yüzlerin basit oluşunu sağlamak için kullanıcının yaptığı eylemin karmaşıklığının gizlenmemesi gerektiğini vurgulamıştır.

Avishai Wool ve diğerleri (2010)'da tespit edildiği üzere, kullanılabilirlik özelliklerinin bazılarını göz ardı etmenin açıkça hatalı bir yaklaşım olduğunu ve güvenlik duvarı kurallarındaki yanlış yapılandırma sorunlarının çözülmesine yardımcı olmadığını belirtmişlerdir.

Voronkov ve diğerleri (2017) güvenlik duvarı yapılandırması, yöneticilerin veya son kullanıcıların günlük işlerinde uğraşmaları gereken karmaşık ve hataya açık bir süreç olduğunu, güvenlik duvarlarının yanlış yapılandırılmasının, ağda çok sayıda güvenlik açığına yol açacağını, bu nedenle güvenlik duvarı yapılandırma sürecinin kullanılabilirlik çalışmalarından büyük ölçüde yararlanabileceğini öne sürmektedir.

Clincy ve Shariar (2018) çalışmasında Web Uygulama Güvenlik Duvarının gerekliliğinin yanı sıra, olumlu veya olumsuz politika tabanlı saldırı algılama modellerinin güçlü ve zayıf yönleri olduğunu belirtmektedir. Web sunucusunun varsayılan yapılandırmasını kullanmanın güvenlik duvarı olmasına rağmen açıklıklara yol açtığını öne sürerek, çözüm olarak ise uygulama güvenlik testleri yapılmasını önermiştir.

Erhan SAYGILI ve Fatih KILIÇ (2019) Web Uygulama Güvenliği Örnek çalışmalarında, Web Uygulama Güvenlik Duvarı (WAF) kullanıldığında web uygulama güvenliği zafiyetlerinin büyük bir kısmının engellendiğini belirtmişlerdir. Diğer siteler incelendiğinde en güçlü güvenliğin dosya yüklemede sağlandığı belirtmişlerdir. Bu sitelerde bu hizmetin bulunmaması veya bu hizmete erişilememesinden kaynaklanabileceğini, en zayıf yönün ise Brute Force zafiyeti olduğu belirtmişlerdir.

Murat ALAGÖZ (2020) Açık Kaynak Web Uygulama Güvenlik Duvarı ModSecurity'nin Kullanılabilirlik Analizi çalışmasında, ModSecurity üzerinde kullanılabilirlik anketleri yaparak, iyileştirmenin sistemin kullanılabilirlik seviyesini arttırdığını, güvenlik duvarı üzerinde doğru kural girme oranını arttırdığını, kullanıcı hataları nedeniyle web sunucusunun hizmet dışı kalma süresini azalttığını tespit edilmiştir. Yardım menüsü ve işletme kılavuzu eksikliği, yetersiz bilgilendirme mesajları, hızlandırıcı kısa yol veya özgün gelişmiş komut satırı bulunmaması ile çalışma kapsamına alınmayan diğer kullanılabilirlik kusurları üzerinde yapılacak iyileştirmelerle ModSecurity'nin kullanılabilirlik düzeyine anlamlı katkılar sağlanabileceğini, böylece yanlış yapılandırmanın azaltılarak uygulama güvenliğinin artırılabileceğini belirtmiştir.

2024 yılında Web Uygulama Güvenlik Duvarı (WAF) alanında önemli bir araştırma KuppingerCole Analysts tarafından gerçekleştirilmiştir. Bu araştırma, WAF pazarının mevcut durumu ve Web Uygulama ve API Koruması (WAAP) yönündeki evrimi üzerine odaklanmaktadır. Raporda, WAF teknolojisindeki gelişmeler, bot yönetimi, API güvenliği entegrasyonu ve anormal aktiviteleri tespit etmek için yapay zeka ve makine öğrenimi kullanımına dair önemli trendler vurgulanmaktadır.

2.MATERYAL VE YÖNTEMLER

Bu çalışmada, Docker programı kullanılarak, zafiyetli web sitesi (DVWA) ve kullandığımız web uygulama güvenlik duvarının, kendisine ait kütüphaneleri ve bağımlılıkları birbirlerini etkilemeden, dış ortamdan izole edilmiş bir şekilde çalıştırılması sağlanmıştır. Kullanılan web uygulama güvenlik duvarı (WAF): ModSecurity'dir.

ModSecurity WAF, tüm WAF'larda olduğu gibi sistemlerin saldırganlar tarafından ele geçirilmesini önlemek için sistemlerimizi koruyan güvenlik duvarlarından biridir. Projemiz kapsamında kullandığımız ModSecurity WAF resmi olmayan geliştiriciler tarafından hazırlanıp DockerHUB gibi depo uygulamalarında paylaşılmıştır.

Güvenlik duvarının test aşamasında, Burp Suite uygulaması kullanılmış ve aynı zamanda manuel saldırılar da yapılmıştır. DVWA üzerinden kullanılan saldırı yöntemleri; Brute Force, CSRF, File Iclusion, File Upload, SQL

Injection, Weak Session IDs, XSS, CSP Bypass'tır. Farklı sızma testi uygulamaları kullanılarak güvenilirlik testi ölçeklendirilmiştir.

2.1 SQL Injection:

SQL (Structured Query Language), veritabanındaki verileri tasarlamayı, yönetmeyi ve düzenlemeyi sağlayan bir programlama dilidir. Ancak, SQL dilindeki açıklıklardan faydalanarak saldırganlar hassas bilgileri ele geçirebilir, veriler üzerinde değişiklik yapabilir, yetkilerini artırabilir veya başka kötü amaçlar için kullanabilir. Bu saldırı tekniğine SQL Injection (SQLi) denir. SQL Injection genellikle web sitelerine yönelik bilinse de, aslında veritabanı olan tüm uygulamalarda ciddi zararlara yol açabilir.

SQL Injection saldırıları genel olarak dört ana sınıfa ayrılabilir:

Klasik SQL Injection: Veritabanına kötü amaçlı SQL komutları enjekte edilerek yapılan saldırıdır.

Blind or Inference SQL Injection: Saldırganın veritabanı ile doğrudan bilgi alması yerine, uygulama yanıtlarından dolaylı yolla bilgi elde ettiği saldırı türüdür.

Veri Tabanı Yönetim Sistemi-Özel SQL Injection: Belirli veri tabanı yönetim sistemlerinin (DBMS) özel yapılarına yönelik yapılan saldırılardır.

Bileşik (Compounded) SQL Injection:

- **SQL Injection + Yetersiz Kimlik Doğrulama**
- **SQL Injection + DDoS Saldırıları**
- **SQL Injection + DNS Korsanlığı**
- **SQL Injection + XSS (Cross-Site Scripting)**

En yaygın SQL Injection saldırısı, hedef internet sitesinin kullanıcı giriş kısmına yönelik yapılır. Böylece, giriş yapan kullanıcı farkında olmadan bilgilerini korsanlara gönderebilir. Bu tür saldırılara karşı önlem almak, oluşabilecek problemleri önlemeye yardımcı olacaktır. Önlemler arasında, uygulamalar tarafından girilen özel karakterlerin kontrol edilmesi ve büyük uygulamaların bu açıklara sahip olup olmadığını test etmek için güvenlik denetimleri yapılması yer alır.

2.2 Brute Force (Kaba Kuvvet Saldırısı):

Brute Force (Kaba Kuvvet) saldırıları, bir saldırganın bir sistem üzerindeki kullanıcı adı veya parolayı doğru tahmin etmek amacıyla sistematik bir şekilde çeşitli kombinasyonları denediği saldırı türüdür. Bu tür saldırılarda, tüm olasılıkları deneyerek doğru kombinasyonu bulmaya çalışılır. Projemiz kapsamında, kaba kuvvet saldırısını zafiyetli web sitemizdeki kullanıcı adı ve parola alanında test ettik.

2.3 XSS (Cross-Site Scripting):

XSS (Cross-Site Scripting), Türkçe adıyla "siteler arası betik çalıştırma," genellikle web uygulamalarında bulunan bir güvenlik açığıdır. XSS saldırıları, OWASP Top 10 listesinde yer alır ve web uygulamalarının güvenliği için ciddi bir risk oluşturur. Bu zafiyete sahip bir sistemde, saldırganlar HTML, CSS ve JavaScript kodları enjekte ederek çeşitli kötü niyetli eylemler gerçekleştirebilirler.

XSS Çeşitleri:

Reflected XSS (Yansıtılmış XSS): Kullanıcıdan alınan verilerin doğrudan ekrana basıldığı durumlarda ortaya çıkar. Reflected XSS sayesinde, kullanıcı tarafından sağlanan veriler kötü niyetli kodlar içerebilir ve bu kodlar doğrudan ekranda çalıştırılabilir. Bu tür bir saldırıda, kullanıcı girişi dinamik olarak işlenir ve ekrana yansıtılır.

Stored XSS (Depolanmış XSS): En tehlikeli XSS türlerinden biridir. Kötü amaçlı bir komut dosyası, doğrudan savunmasız bir web uygulamasının veri tabanına veya depolama alanına enjekte edilir ve burada saklanır.

Depolanmış XSS, diğer kullanıcıların bu verileri görmesi ve etkileşimde bulunmasıyla aktif hale gelir, bu da saldırının geniş çaplı etkiler yaratmasına neden olabilir.

DOM-based XSS (DOM Tabanlı XSS): Uygulamanın, güvenilmeyen kaynaklardan gelen verileri doğrudan Document Object Model (DOM) içindeki potansiyel olarak tehlikeli bir hedefe yazarak güvenli olmayan bir şekilde işlemesiyle ortaya çıkar. Bu tür XSS, istemci tarafı JavaScript kodları aracılığıyla gerçekleştirilir ve DOM'daki verileri manipüle ederek kötü niyetli kodların çalışmasına neden olabilir.

2.4 CSRF (Cross-Site Request Forgery):

Kurumsal ve standart kullanıcılar tarafından kullanılan web uygulamaları genellikle kişisel verilerle ve kimlik bilgileriyle korunur. Web uygulamaları ve tarayıcılar arasında taşınan çerezler (cookies) bu kimlik bilgilerini içerdiğinden, güvenlik açısından büyük önem taşır.

CSRF (Siteler Arası İstek Sahteciliği), bir web sitesindeki güvenlik açığını kullanarak, bir kullanıcının kimliğini taklit ederek yetkisiz işlemler yapma sürecidir. Saldırgan, kullanıcının kimlik bilgilerini kullanarak, kullanıcının izni olmadan, web uygulamasında işlem yapar. Bu tür saldırılar genellikle GET istekleri ve oturum yönetimi hatalarından yararlanır.

CSRF, OWASP Top 10 listesinde yer alan ve en sık karşılaşılan çevrimiçi saldırılardan biridir. Bu zafiyet, popüler web uygulamalarında bile sıkça görülebilir. Saldırganlar, genellikle kurbanı sahte bir istek göndermesi için kandıran kötü niyetli e-postalar veya bağlantılar kullanarak CSRF saldırıları gerçekleştirir. Şüphelenmeyen kullanıcı, saldırı sırasında kimliğinin doğrulandığını fark etmeden meşru bir talepte bulunur ve bu, sahte isteklerle birleştirilir.

Projemizde, CSRF saldırısını zafiyetli web sitemizdeki kullanıcı adı ve parola alanlarında test ettik. Kullanıcı bilgilerini değiştirdikten sonra, açık olan şifre değiştirme bağlantısını kullanarak saldırıyı simüle ettik. Bu yöntemle, CSRF saldırısının etkilerini ve güvenlik açıklarını değerlendirdik.

CSRF Nasıl Gerçekleşir?

CSRF (Siteler Arası İstek Sahteciliği) saldırısı, bir web uygulamasındaki oturum çerezi (cookie) ve kimlik bilgilerini kullanarak yetkisiz işlemler yapmayı hedefler. Saldırı, genellikle kurbanın kimlik bilgileri doğrulanmış bir durumda olduğu bir senaryoda gerçekleşir.

Saldırgan, kötü niyetli bir bağlantı, e-posta veya web sayfası aracılığıyla kurbanı kandırarak, sahte bir istek gönderir. Kurban, bu bağlantıya tıkladığında veya kötü niyetli içeriği etkileşimde bulunduğunda, saldırganın oluşturduğu istek, kurbanın kimlik bilgilerini ve oturum çerezlerini kullanarak hedef web uygulamasında yetkisiz işlemler yapar. Örneğin, bir bankacılık sistemi açık bir oturumdayken, kurbanın e-posta adresine gönderilen tehlikeli bir bağlantıya tıklanması durumunda, saldırgan bankacılık sistemine erişim sağlayarak para transferi gibi işlemler gerçekleştirebilir.

CSRF saldırıları genellikle bankacılık, sosyal medya ve ağ cihazları gibi web arayüzlerine karşı yapılır ve bu uygulamalarda ciddi güvenlik açıklarına yol açabilir.

2.5 CAPTCHA Kullanımı

Bir web formunda captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) bilgisi doğru girilmediği sürece işlem gerçekleştiremeyeceği için "CSRF" saldırısına karşı alınacak bir önlem niteliğindedir.

2.6 File Inclusion:

Saldırganın, hedef web sitesine bir dosya dahil etmesine veya hedef web sitesinin içerisinde bulunan fakat sunulmamış bir dosyayı görüntüleyebilmesine file inclusion denir. Bu çalışmada, DVWA (Damn Vulnerable Web Application) adlı web uygulamasının bir güvenlik zafiyetinden faydalanarak File Inclusion saldırısı gerçekleştirilmiştir.

Local File Inclusion Nasıl Yapılır?

Local File Inclusion saldırısı, hedef sitenin barındığı sunucudaki ziyaretçilere sunulmamış dosyanın hedef site üzerinden görüntülenebilmesine denir. Öncelikle Mod Security isimli web güvenlik duvarını Docker üzerinde çalıştırıp DVWA'nın önüne kurduktan sonra DVWA'nın bize sunduğu seçeneklerden File Inclusion kısmı açılmalıdır.

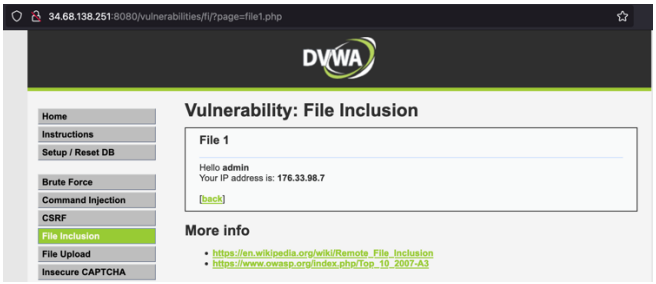
Şekil.2.6.1. File Inclusion

Ardından tarayıcının adres çubuğunda görünen linke bakılmalıdır:

http://34.68.138.251/vulnerabilities/fi/?page=include.php linkini bu işlemi yaparken kullandığınız bilgisayarın IP adresi doğrultusunda görülmektedir.
Yukarıdaki linkte "?" işareti parametrelerin sıralanacağı kısmın başını ifade eder. "=" işareti parametreye değer atanacağını ifade eder. Yukarıdaki linkte bir tane parametre ve bir de onun değeri mevcuttur. Bu parametrenin ismi page, yani sayfadır. Değeri ise bir dosya ismidir.

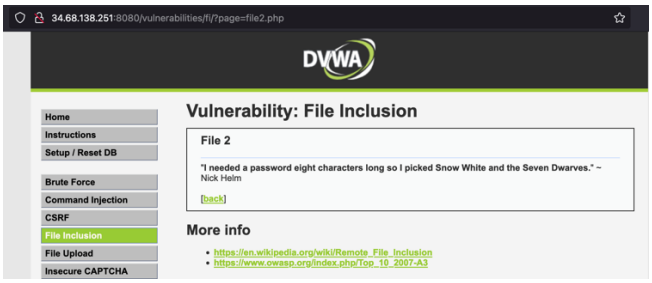
DVWA'nın bize sunduğu [file1.php], [file2.php] ve [file3.php] linklerine sırayla tıklayalım ve linkteki değişimi ve içerikteki değişimi gözlemleyelim:
Adım 1:

[file1.php] 'e tıklandığında:



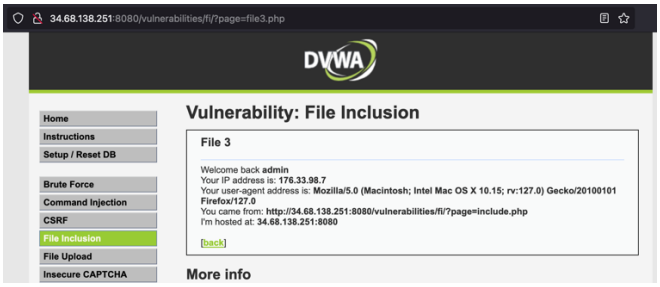
Şekil.2.6.2. file1.php

[back] butonuna tıklandığında tekrar file seçim kısmına dönüş yapılmıştır.
Adım 2:
[file2.php] 'ye tıklandığında:



Şekil.2.6.3. file2.php

[back] butonuna tıklandığında yeniden file seçim kısmına dönüş yapılmıştır.
Adım 3:
[file3.php] 'e tıklandığında:



Şekil.2.6.4. file3.php

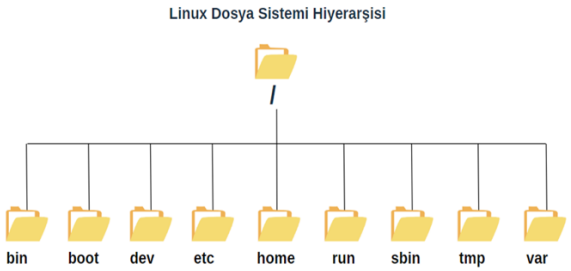
Yukarıda görüldüğü üzere page parametresi sırayla değişik üç adet dosya ismi almaktadır ve aldığı değerlere göre içerik değiştirmektedir. Yani linkteki page parametresi, içerik olarak yansıtılacak php dosyasını belirleyen bir dosya seçici olarak kullanılmaktadır. Bu seçim sonucunda mevcut sayfanın barındığı

http://34.68.138.251/vulnerabilities/fi/

dizinindeki bir dosya, index.php'ye görüntülenen File Inclusion sayfasına dahil ediliyor. Bu mekanizmaya File Inclusion, yani Dosya Dahil Etme denmektedir.

Şimdi saldırı aşamasına geçelim:

Öncelikle saldırı için Linux sistemlerin dosya hiyerarşisine bakalım:



Şekil.2.6.5. Linux Dosya Hiyerarşisi

Yukarıda da gördüğümüz, bizim bulunduğumuz

/var/www/dvwa/vulnerabilities/fi/

dizininden var klasörüne geçebilmek için birkaç kez üst dizine çıkılması gerekir. Var klasörüne varana kadar üst dizine ulaşılmalıdır.

Beş tane ../ komutundan kullanılırsa var dizinine ulaşılır. Şimdi yapılması gereken sayfa linkine tıklayıp page parametresine değer olarak
../../../../../

eklenirse fi klasörünün içinden çıkıp var klasörünün yer aldığı klasör havuzuna erişilir. Sıradaki işlem var klasörünün sibling'i (kardeşi) olan etc klasörüne dallanmaktır. Bunun için /etc/ dizini eklenir ve sonrasında etc klasörü içerisinde barınan dosyanın ismi eklenir:

../../../../../etc/passwd

35.192.76.2/vulnerabilities/fi/?page=../../../../../etc/passwd

Page parametresine yukarıdaki komutu eklediğimizde DVWA'ya kurduğumuz WAF'ın engellemesini beklenilir. Elde edilen sonuçlar bulgular kısmında belirtilmiştir.

Remote File Inclusion (RFI)

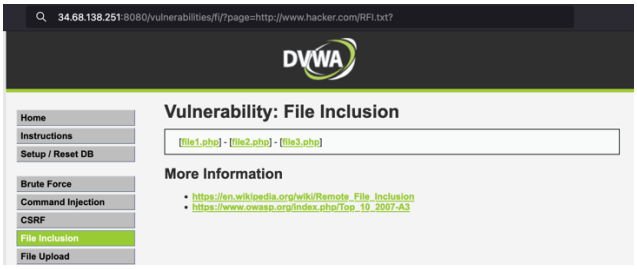
RFI, uzaktan dosya dahil etmek anlamına gelir. Bu saldırı türünde, saldırgan hedef siteye kendi dosyasını (örneğin, bir shell dosyasını) dahil ederek görüntüler. Bu açık, diğer açık türlerinde olduğu gibi izinsiz olarak yapılır. RFI'nin mantığı, açık bulunan siteye saldırganın istediği bir dosyayı dahil etmektir. İşleyişi LFI (Local File Inclusion) ile benzerdir, ancak farkı, ekrana dahil edilecek dosyanın hedef sitenin sunucusunda yer alan bir dosya değil, harici bir dosya (saldırganın kendi dosyası) olmasıdır. Saldırgan, shell dosyasını yükleyerek sisteme sızabilir ve kötü amaçlı faaliyetlerde bulunabilir.

Remote File Inclusion Nasıl Yapılır?

DVWA üzerinden

http://35.192.76.2/vulnerabilities/fi/?page=http://www.hacker.com/RFI.txt?

şeklinde çağıralım.
Burada DVWA'da linkte bulunan page parametresine http://www.hacker.com/RFI.txt? komutu atanmıştır. Eğer sayfa üzerinde çağrılan dosya çalıştıysa açık bulunmuştur ve siteye shell yüklenmiştir.



Şekil 2.6.6. .Remote File Inclusion

2.7 File Upload:

File Upload, bir web uygulaması üzerinde bulunan dosya yükleme kısmında sunucuya zararlı bir yazılım yükleyerek gerçekleştirilen saldırılara verilen genel addır. File Upload saldırılarının gerçekleşmesinin sebebi, dosyaları yüklenilen bölümde filtreleme algoritmasının yanlış ya da eksik olmasıdır. Bu şekilde hatası olan bir web sayfasında kullanıcı kendi profil fotoğrafını değiştirmek istediğinde "Dosya Yükle" kısmına zararlı bir php

dosyası seçilirse sisteme erişilebilir. Bu da bir güvenlik açığı oluşturur. Zafiyetli web sitesinde, kullanıcıdan bir resim dosyası yüklenmesi beklenmektedir.Yüklenen resim dosyasının yeri de ekrana basılmaktadır. Bu bilgiye göre dosya, mevcut dizinin iki üst dizininde bulunan "hackable/uploads" dizinine kaydedilmektedir. Şekil 2.7.1'de kaydedildiğini görmektesiniz.

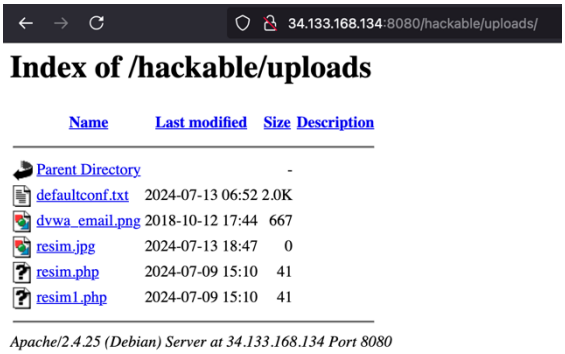
Şekil 2.7.1. ../../hackable/uploads/Resim.jpg succesfully uploaded!

Yukarıda kırmızı renkle verilen bildirimde, yüklenen dosyanın konumunu verilmektedir. Bu linke gidip ilk yazılan komutun sağladığı "cmd=" yapısıyla iki dosya geriye gidilirse dvwa dizinine ulaşılır. Buradan da hackable dizinin altında olan uploads dizinine ulaşılır.

Kaydedilen resim ve resmin kaydedildiği dizin aşağıdaki gibidir.

0.0.0.0:49154/hackable/uploads/

Şekil 2.7.2.'de ***0.0.0.0:49154/hackable/uploads/*** adresini görmektesiniz.

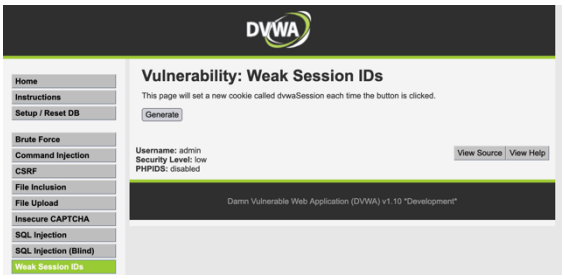


Şekil 2.7.2. File Upload için denenen görseller

2.8 Weak Session IDs

Oturum kimlikleri (daha sık bilinen adıyla Session IDs), bir sistemde kayıtlı olan kullanıcıların ağ iletişimlerinde kullandıkları bir veridir. Bu veri sayesinde kullanıcının kim olduğu anlaşılır ve daha önceki verilerin sunucuda tutularak yapılan işlemlerin kaydının alınması sağlanır. Zayıf oturum kimlikleri ise kullanıcılarınızın oturumlarının ele geçirilmesine neden olur. Oturum kimlikleri eğer küçük değer aralıklarından seçilirse tahmini kolay olur ve saldırganların bu kimlikleri ele geçirmesi kolaylaşır.

Kullandığımız web yazılımı olan DVWA'ın Weak Session IDs kısmının görüntüsü aşağıdaki şekildedir:



Şekil 2.8.1. .Weak Session IDs Ana Ekran Görüntüsü

3.BULGULAR

3.1 SQL Injection Bulguları:

Aşağıda proje kapsamında kullanılan ModSecurity WAF'ın başarılı bir şekilde çalışmasına izin vermeyen payloadlara örnekler verilmiştir;

- 1 AND (SELECT * FROM Users) = 1
- un?<ion sel="">+un/**/ion+se/**/lect+
- " or sleep(5)#
- AND 1083=1083 AND (1427=1427

Bu payloadlara karşı WAF'ın gösterdiği tepkiler loglar üzerinden şu şekilde gözlemlenmiştir;

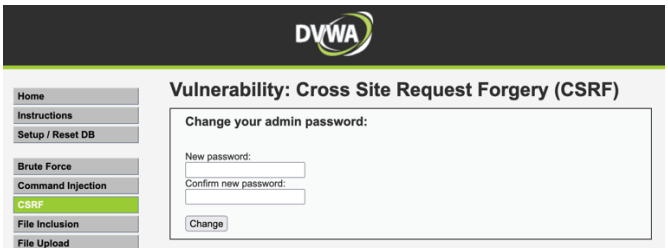
Şekil 3.1.1. .SQL Injection

Aşağıda web uygulama güvenlik duvarının engellediği payloadlara örnekler verilmiştir:

- OR 1=1
- AND 7300=7300 AND ('pKlZ'='pKlY
- and (select substring(@@version,3,1))='c'
- /?id=1+OR+0x50=0x50 /?

3.2 CSRF Bulguları:

Web uygulamasını kullanmakta olan kullanıcıların istekleri dışında işlemler yürütülmesidir. Örnek olarak CSRF'te parola değişimi yapılabilmektedir. CSRF sayfasına gelindiğinde Şekil.17 ekranı görüntülenmektedir.

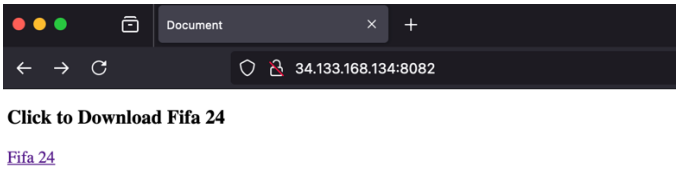


Şekil 3.2.1. CSRF Ekranı

Bu sayfayla kurbanı kandırarak şifresi değiştirebilir.

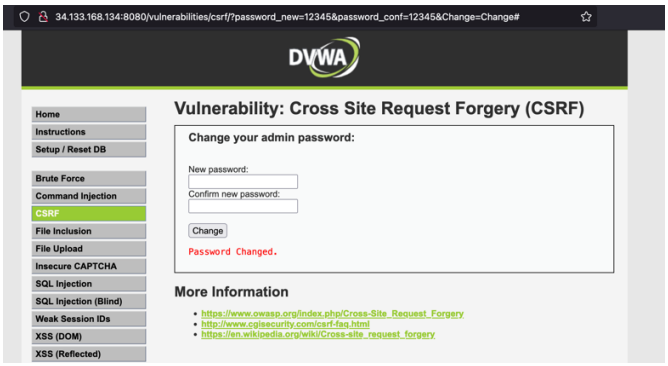
Bu ekrana rastgele bir şifre girildiğinde /vulnerabilities/csrf/?password_new=31231&password_conf=1231231&Change=Change# böyle bir URL ile karşılaşıldı.

Bu URL'i manipüle etmek amacıyla sahte bir website oluşturulduğunda:



Şekil 3.2.6 Sahte web sitesi

Kurban, siteye tıkladığı zaman şifresi değişecektir.



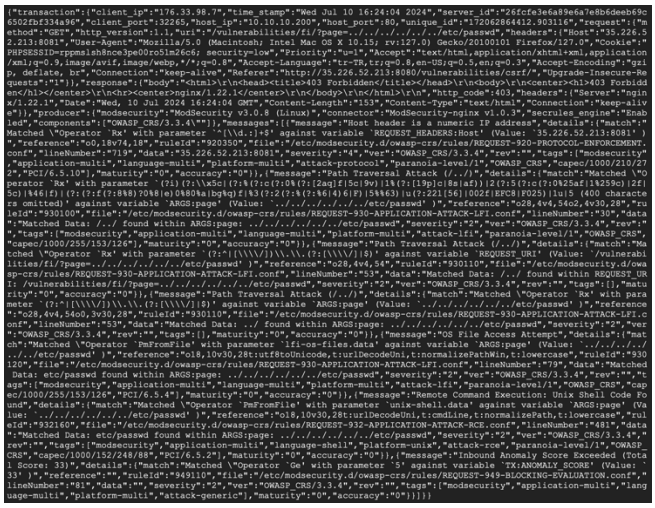
Şekil 3.2.6 Sahte web sitesi

Ve şifre değişti. Aynı tekniğin XSS stored da bağlantısını

<img src="http://0.0.0.0/vulnerabilities/csrf/?password_new=admin &password_conf=admin&Change=Change#" width="0" height="0"> olarak ekleyip tüm kullanıcıların şifresi değiştirilebilir. Bu isteği WAF engellememektedir.

3.3 FILE INCLUSION Bulguları:

RFI VE LFI saldırıları için WAF %100 koruma göstermiştir.

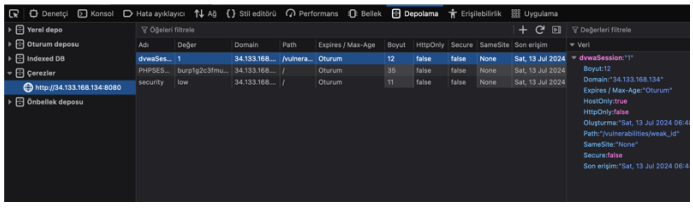

{"transaction":{"client_ip":"176.33.98.7","time_stamp":"Wed Jul 10 16:24:04 2024","server_id":"26fcfe3e6a89e6a7e8b6deeb69c6502fbf334a96","client_port":32265,"host_ip":"10.10.10.200","host_port":80,"unique_id":"1720628644412.903116","request":{"method":"GET","http_version":1.1,"uri":"/vulnerabilities/fi/?page=../../../../../../etc/passwd","headers":{"Host":"35.226.52.213:8081","User-Agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:127.0) Gecko/20100101 Firefox/127.0","Cookie":"PHPSESSID=rppns1ah8eca3pe00ro1m26c6j; security=low","Priority":"u=1","Accept":"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8","Accept-Language":"tr-TR,tr;q=0.8,en-US;q=0.5,en;q=0.3","Accept-Encoding":"gzip, deflate, br","Connection":"keep-alive","Referer":"http://35.226.52.213:8080/vulnerabilities/csrf/","Upgrade-Insecure-Requests":"1"}},"response":{"body":"<html>\r\n<head><title>403 Forbidden</title></head>\r\n<body>\r\n <h1>403 Forbidden</h1> \r\n<hr> nginx/1.22.1 \r\n</body>\r\n</html>\r\n","http_code":403,"headers":{"Server":"nginx/1.22.1","Date":"Wed, 10 Jul 2024 16:24:04 GMT","Content-Length":"153","Content-Type":"text/html","Connection":"keep-alive"}},"producer":{"modsecurity":"ModSecurity v3.0.8 (Linux)","connector":"ModSecurity-nginx v1.0.3","secrules_engine":"Enabled","components":["OWASP_CRS/3.3.4\""]},"messages":[{"message":"Host header is a numeric IP address","details":{"match":"Matched \"Operator `Rx' with parameter `(\\d.:]+$' against variable `REQUEST_HEADERS:Host' (Value: `35.226.52.213:8081')","reference":"o0,18v74,18","ruleId":"920350","file":"/etc/modsecurity.d/owasp-crs/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf","lineNumber":"719","data":"35.226.52.213:8081","severity":"4","ver":"OWASP_CRS/3.3.4","rev":"","tags":["modsecurity","application-multi","language-multi","platform-multi","attack-protocol","paranoia-level/1","OWASP_CRS","capec/1000/210/272","PCI/6.5.10"],"maturity":"0","accuracy":"0"}},{"message":"Path Traversal Attack (/../)","details":{"match":"Matched \"Operator `Rx' with parameter `(?i)(?:\\x5c|(?:%(?:c[0-8]%(?:[2aq]f|5c|9v)|1%(?:[19p]c|8[ab]))|2(?:5(?:c(?:0%25af|1%259c)|2f|5c)|46|f)|(?:(?:f(?:8[89]|e[0-8])|80%a[0-9a-f])f|%3(?:2(?:%(?:%6(4|6)|[0-5][0-9a-f])|[0-46-9a-f])|[0-9a-f])))' against variable `ARGS:page' (Value: `../../../../../etc/passwd')","reference":"o28,4v4,54o2,4v30,28","ruleId":"930100","file":"/etc/modsecurity.d/owasp-crs/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf","lineNumber":"30","data":"Matched Data: /../ found within ARGS:page: ../../../../../../etc/passwd","severity":"2","ver":"OWASP_CRS/3.3.4","rev":"","tags":["modsecurity","application-multi","language-multi","platform-multi","attack-lfi","paranoia-level/1","OWASP_CRS","capec/1000/255/153/126"],"maturity":"0","accuracy":"0"}},{"message":"Path Traversal Attack (/../)","details":{"match":"Matched \"Operator `Rx' with parameter `(?:^|[\\\\/])\\.\\.(?:[\\\\/]|$)' against variable `REQUEST_URI' (Value: `/vulnerabilities/fi/?page=../../../../../etc/passwd')","reference":"o28,4v4,54","ruleId":"930110","file":"/etc/modsecurity.d/owasp-crs/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf","lineNumber":"53","data":"Matched Data: /../ found within REQUEST_URI: /vulnerabilities/fi/?page=../../../../../../etc/passwd","severity":"2","ver":"OWASP_CRS/3.3.4","rev":"","tags":[],"maturity":"0","accuracy":"0"}},{"message":"Path Traversal Attack (/../)","details":{"match":"Matched \"Operator `Rx' with parameter `(?:^|[\\\\/])\\.\\.(?:[\\\\/]|$)' against variable `ARGS:page' (Value: `../../../../../etc/passwd')","reference":"o28,4v4,54o0,3v30,28","ruleId":"930110","file":"/etc/modsecurity.d/owasp-crs/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf","lineNumber":"53","data":"Matched Data: ../ found within ARGS:page: ../../../../../../etc/passwd","severity":"2","ver":"OWASP_CRS/3.3.4","rev":"","tags":[],"maturity":"0","accuracy":"0"}},{"message":"OS File Access Attempt","details":{"match":"Matched \"Operator `PmFromFile' with parameter `lfi-os-files.data' against variable `ARGS:page' (Value: `../../../../../etc/passwd')","reference":"o18,10v30,28t:utf8toUnicode,t:urlDecodeUni,t:normalizePathWin,t:lowercase","ruleId":"930120","file":"/etc/modsecurity.d/owasp-crs/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf","lineNumber":"79","data":"Matched Data: etc/passwd found within ARGS:page: ../../../../../../etc/passwd","severity":"2","ver":"OWASP_CRS/3.3.4","rev":"","tags":["modsecurity","application-multi","language-multi","platform-multi","attack-lfi","paranoia-level/1","OWASP_CRS","capec/1000/255/153/126","PCI/6.5.4"],"maturity":"0","accuracy":"0"}},{"message":"Remote Command Execution: Unix Shell Code Found","details":{"match":"Matched \"Operator `PmFromFile' with parameter `unix-shell.data' against variable `ARGS:page' (Value: `../../../../../etc/passwd')","reference":"o18,10v30,28t:urlDecodeUni,t:cmdLine,t:normalizePath,t:lowercase","ruleId":"932160","file":"/etc/modsecurity.d/owasp-crs/rules/REQUEST-932-APPLICATION-ATTACK-RCE.conf","lineNumber":"481","data":"Matched Data: etc/passwd found within ARGS:page: ../../../../../../etc/passwd","severity":"2","ver":"OWASP_CRS/3.3.4","rev":"","tags":["modsecurity","application-multi","language-shell","platform-unix","attack-rce","paranoia-level/1","OWASP_CRS","capec/1000/152/248/88","PCI/6.5.2"],"maturity":"0","accuracy":"0"}},{"message":"Inbound Anomaly Score Exceeded (Total Score: 33)","details":{"match":"Matched \"Operator `Ge' with parameter `5' against variable `TX:ANOMALY_SCORE' (Value: `33')","reference":"","ruleId":"949110","file":"/etc/modsecurity.d/owasp-crs/rules/REQUEST-949-BLOCKING-EVALUATION.conf","lineNumber":"81","data":"","severity":"2","ver":"OWASP_CRS/3.3.4","rev":"","tags":["modsecurity","application-multi","language-multi","platform-multi","attack-generic"],"maturity":"0","accuracy":"0"}}]}

Şekil 3.3.2. LFI BurpSuite Ekran Çıktısı

Local File Inclusion saldırısı gerçekleştirilmiş olundu. ModSecurity WAF'ının Local File Inclusion saldırısını geçirmediği görülmüştür. ModSecurity bu saldırıyı engellemektedir.

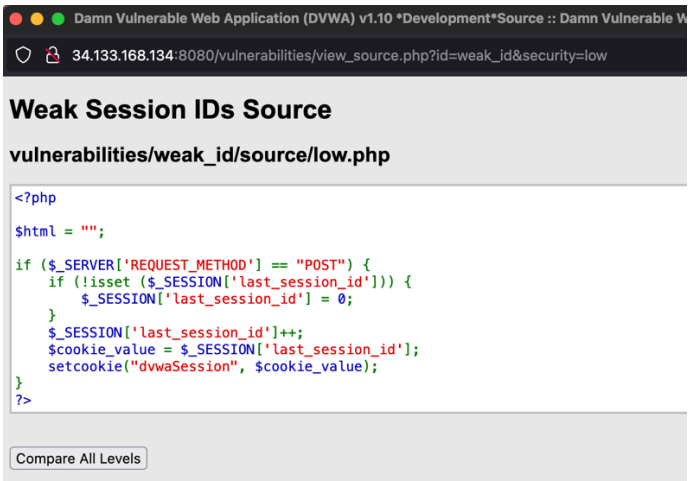
3.4 WEAK SESSION IDs Bulguları:

Sitenin kaynaklarından, Session ID'ye ulaşılmak istenildiği için Storage kısmına girilir.



Şekil 3.4.1. Storage

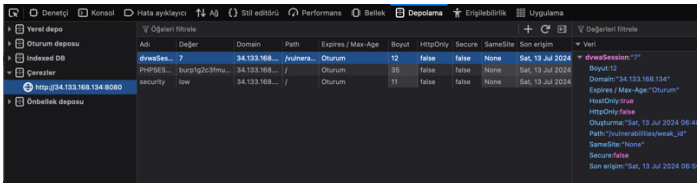
Weak Session ID kısmının kaynak kodları bu şekildedir:



Şekil 3.4.2. .Weak Session ID Kaynak Kodları

Kaynak kodlardan da anlaşılacağı üzere eğer eski bir Oturum Kimliği ID'si yoksa bu veriyi direkt 0 olarak başlatılıp her "generate" butonuna basıldığında Oturum Kimliğinin değeri 1 artmaktadır.

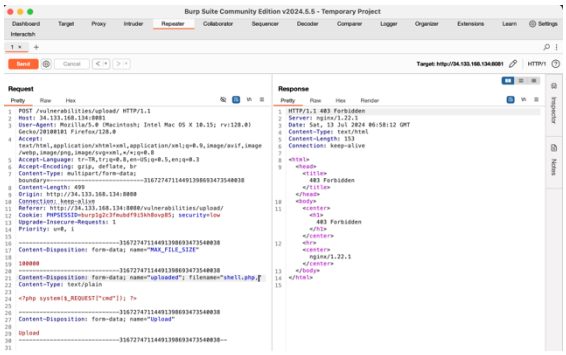
Sistemde ulaşılan veri aşağıdaki gibidir:



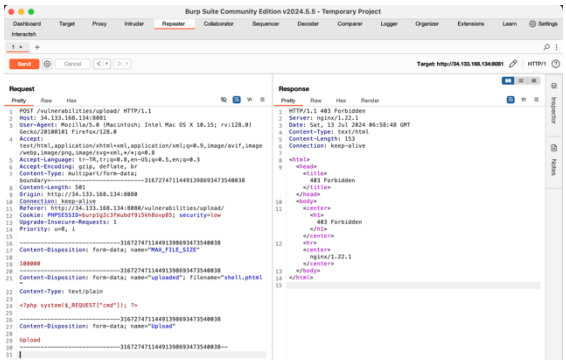
Şekil 3.4.3. Weak Session ID

3.5 FILE UPLOAD Bulguları:

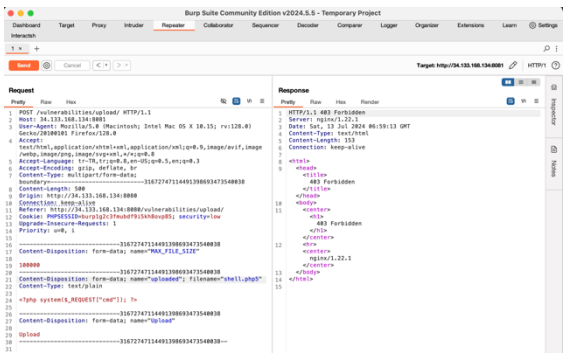
Burp Suite uygulaması üzerinden File Upload saldırısının çıktıları:



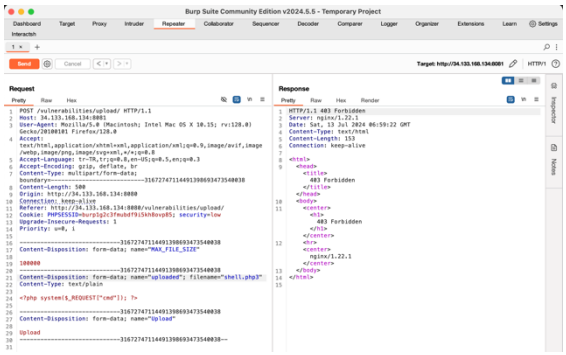
Şekil 3.5.1.Burp Suite File Upload Ekranı 1



Şekil 3.5.2.Burp Suite File Upload Ekranı 2



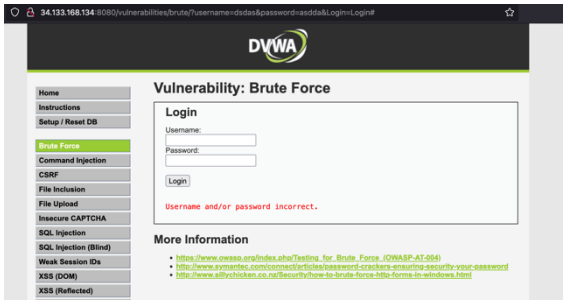
Şekil 3.5.3.Burp Suite File Upload Ekranı 3



Şekil 3.5.4.Burp Suite File Upload Ekranı 4

File upload saldırısı yapılan zafiyetli web sitesi üzerinde Burp Suite kullanılarak çıktılar elde edilmiştir. Çıktılarda görüldüğü üzere File Upload sadırıları engellenmiştir.

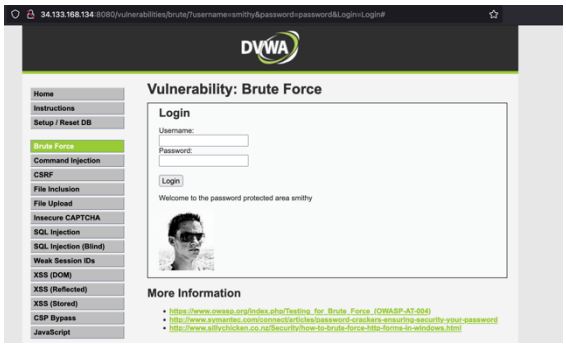
3.6 BRUTE FORCE Bulguları:



Şekil 3.6.1.Yanlış Şifre Girildiğinde DVWA Ekran Çıktısı

Şekil 3.6.2. Yanlış şifre girildiğinde ModSecurity WAF Çıktıları

Denemeler sonucu doğru şifre girildiğinde sistem saldırıyı engellememiştir.

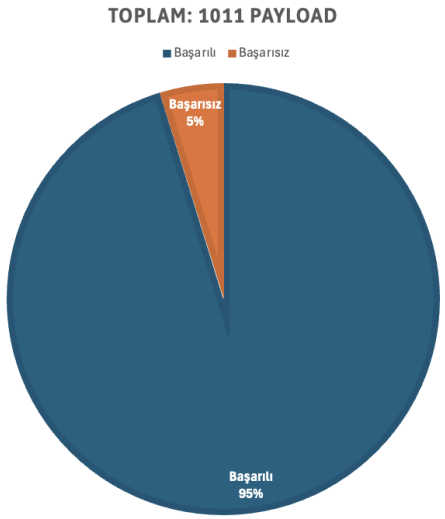


Şekil 3.6.3.Deneme sonucu

4.SONUÇLAR VE TARTIŞMA

4.1 SQL INJECTION Sonuçları

ModSecurity üzerinde BurpSuite uygulaması kullanılarak yapılan SQLi ataklarının sonuçları yukarıdaki tabloda görülmektedir. Elde edilen sonuçlara göre ModSecurity WAF'ının saldırı için kullanılan payloadlara karşı başarılı/başarısız olduğunu gösteren yüzde grafiği aşağıdaki şekilde gösterilmiştir.



Şekil 4.1.1. ModSecurity

ModSecurity WAF'ının güvenliğini sağladığı zafiyetli web sitesinde SQL Injection için toplam 1011 payload denenmiştir. Elde edilen sonuçları aşağıdaki tablodan görebilirsiniz:

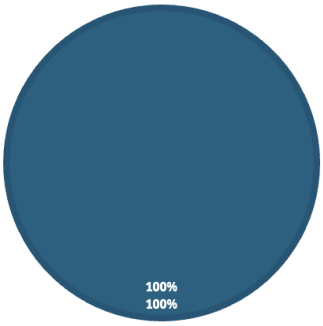
TOPLAM PAYLOAD SAYISI	WAF Passed	WAF Blocked	YÜZDE
1011	48	963	%4,74

Şekil 4.1.2. SQL Injection WAF Sonuçları

PAYLOAD	WAF Blocked
‘	+
‘’	+
OR 1=1#	+
OR 1=1--	+
ORDER BY 1 --	+
ORDER BY 1#	+
AND 7300=7300 AND ('pKlZ'='pKlY	+
AND 7300=7300 AND ('pKlZ'='pKlZ	+
RLIKE (SELECT (CASE WHEN (4346=4347) THEN 0x61646d696e ELSE 0x28 END)) AND 'Txws'='	+
RLIKE (SELECT (CASE WHEN (4346=4346) THEN 0x61646d696e ELSE 0x28 END)) AND 'Txws'='	+
and (select substring(@@version,1,1))='X'	+
and (select substring(@@version,3,1))='c'	+
AND 1083=1083 AND (1427=1427	-
admin’ #	-
admin” #	-
or sleep(5)#	-

Şekil 4.1.3. SQL Injection Genel Sonuç Tablosu (+: WAF Başarılı)

4.2 FILE INCLUSION Sonuçları:



Şekil 4.2.1. File Inclusion

Local File Inclusion (LFI) ve Remote File Inclusion (RFI) saldırıları, ModSecurity web güvenlik duvarı üzerinde test edilmiştir. DVWA üzerindeki saldırı araçları ve Burp Suite kullanılarak gerçekleştirilen saldırı denemelerinde, WAF %100 koruma sağlamıştır.

File inclusion saldırılarından korunmak için web sitelerinin kodları daha güçlü hale getirilmelidir. Kullanıcıdan alınan veriler ve giriş alanları filtrelenmeli ve bu veriler değişkenlere atanırken dikkatli olunmalıdır. Ayrıca, PHP yapılandırması (php.ini) doğru bir şekilde yapıldığında bu tür saldırılara karşı korunma olasılığı artar.

4.3 CSRF Sonuçları:

Kullanılan WAF, bağlantı şifre değiştirmeye izin vermiş veya XSS'e gömülmüş resim olarak yüklenmiş belgeyi engellememiştir. Bu nedenle, WAF kullanmak isteyen kullanıcılar ek güvenlik önlemleri olarak CSRF token ve özel kurallar kullanmalıdırlar.

CSRF Zafiyetinde Alınabilecek Önlemler:

1. **Sistem Taraflı Alınabilecek Önlemler:**
    - **Token Kullanımı:** Kullanıcıya her oturum için rastgele ve benzersiz "token" bilgisi verilir.

Örnek Senaryo:

Aşağıda verilen kodda, saldırgan img etiketini kullanarak bağlantıyı bir resim bağlantısı olarak gösterip sisteme daha önce kayıt olmuş bir kullanıcı gibi giriş yapmak istemektedir:

```
<img src="abc.php?giris=dogru">
```

Bu tür açıkları kapatmanın en pratik yolu token kullanımıdır:

```
<?php

session_start();

$_SESSION["token"] = sha1(rand());

echo '<a href="abc.php?giris=dogru&token=' . $_SESSION["token"] .
'">Link</a>';

?>
```

abc.php dosyasındaki session kontrolü şu şekilde olmalıdır:

```
<?php session_start(); if ($_GET["giris"] == "dogru") { if (isset($_GET["token"])
&& $_GET["token"] == $_SESSION["token"]) { // Doğruysa oturumu başlat echo "Oturum
başlatıldı"; } else { echo "Token yanlış!"; } } ?>
```

2. **GET Metodu Yerine POST Metodu Kullanımı:** Kullanıcıdan alınan önemli veriler POST metodu ile alınmalıdır.

4.4 CAPTCHA Sonuçları:

Web uygulamalarında veri ve cookie bilgileri düzenli olarak silinmelidir. Kişisel bilgilerin bulunduğu web uygulamalarında oturum bilgileri bilgisayarda kaydedilmemelidir. Ayrıca, bilinmeyen e-posta ve bağlantılara karşı dikkatli olunmalıdır.

4.5 BRUTE FORCE Sonuçları:

ModSecurity WAF üzerinde manuel olarak kullanıcı adı ve şifre denemesi yapıldı. Aynı kullanıcı adıyla yapılan 15 farklı şifre denemesinde sistem, yeni kullanıcı adı ve şifre girişlerini engellememiştir. Bu durum, kullanılan WAF'ın girilen parolalara tepki verip engelleme yapmadığını göstermektedir. Bu nedenle, Brute Force saldırılarını önlemek isteyen yöneticiler, WAF'ın yanı sıra sistemlerinde CAPTCHA, iki faktörlü kimlik doğrulama ve giriş deneme sayısını sınırlama gibi ek güvenlik önlemleri kullanmalıdır. Ayrıca, kullanıcılar da daha güvenli şifreler oluşturmak için uzun, küçük ve büyük harf kombinasyonları ile özel karakterler içeren şifreler kullanmalıdır.

4.6 WEAK SESSION IDs Sonuçları:

Şekil 3.4.6'da görüldüğü gibi, web sayfasının kaynak kodlarındaki zayıflık, oturum kimliklerine erişimi kolaylaştırmıştır. ModSecurity web uygulama duvarları, bu zayıf oturum kimlikleri üzerinde herhangi bir koruma sağlamamaktadır. Web uygulama duvarlarını kullanan geliştiriciler, sistem kaynak kodlarını yazarken daha güçlü kimlik alanları seçmeli ve oturum kimliklerini korumak için ek güvenlik önlemleri uygulamalıdır.

4.7 FILE UPLOAD Sonuçları:

Saldırganlar, dosya uzantısını bir proxy aracılığıyla değiştirerek tarayıcı tabanlı filtrelemeyi atlatmaya çalışabilirler. Bu nedenle, dosya uzantısı kontrolünün sadece web tarayıcısında değil, aynı zamanda sunucuda da yapılması bu tür saldırılardan korunmak için kritik öneme sahiptir.

File upload saldırılarını önlemek için yapılabilecekler:

- Yalnızca belirli dosya türlerine izin verilmelidir.
- Yüklenen dosyaları kötü amaçlı yazılım için taranmalıdır.
- Olası gömülü tehditler tespit edilmelidir.
- Kullanıcılardan dosya yüklemeden önce kimlik doğrulaması istenmelidir.
- Yüklenen dosyalar için maksimum ve minimum boyut sınırları belirlenmelidir.
- Kullanıcıların yüklediği dosya isimlerini rastgele değiştirerek, sisteme dosya yükleyen kullanıcıların erişimi engellenmelidir.

4.8 XSS Sonuçları:

Yapılan test ve saldırılarda kullanılan WAF ile elde edilen sonuçlar Şekil 4.8.1 üzerinde gösterilmiştir.

PAYLOAD	**WAF Blocked**
";a=prompt,a()//	-
';a=prompt,a()//	-
<script>alert(1)<!	+
'-eval("window['pro'%2B'mpt'](8)")-'	+
<img/src/onerror=prompt(8)>	+
</script><svg onload=alert(1)>	+
/alert(1)">'onload="/<svg/1='	+
test	+
‘;’;;!—“;<;XSS>;=&;{()}	-
<;br size=\”;&;{alert(‘XSS’)}\”;>;	-
<A HREF=”http://1113982867/";>XSS</A>	+

Şekil 4.8.1. XSS Sonuç Tablosu (+: WAF Başarılı)

Zafiyetli web sitesinin önüne kurulan web güvenlik duvarına Burp Suite uygulaması kullanılarak yapılan XSS saldırılarına web güvenlik duvarının tepkisi Şekil 4.8.2 ve Şekil 4.8.3 üzerinde görülmektedir.

TOPLAM PAYLOAD SAYISI	WAF Passed	WAF Blocked	YÜZDE
6613	45	6568	%0,68

Şekil 4.8.2. WAF Sonuçları

Kullanılan 6613 payload sonucu ModSecurity WAF %99,32 başarı oranı elde etmiştir.

4.9. Genel Sonuçlar:

SALDIRI TÜRÜ	WAF Engelleme Oranı
SQL INJECTION	%95,26
XSS	%99,32
FILE INCLUSION	%100
BRUTE FORCE	%0
WEAK SESSION Ids	%0

Şekil 4.9.1. Genel Sonuç Tablosu

Bilişim teknolojilerinde sistem ve yazılım güvenliği, günümüzde vazgeçilmez bir unsur haline gelmiştir. Kurumlar, ağlarını korumak ve güvenliğini sağlamak için güvenlik duvarları kullanmaktadır. Bu güvenlik duvarları, hem internet üzerinden gelen tehditlere karşı hem de kurum içinden kaynaklanan kötü niyetli saldırılara karşı sistemleri korur ve kurum içi yönetimde büyük kolaylık sağlar. Bu çalışmada, zafiyetler barındıran bir web uygulamasına (DVWA) korumalı ve korumasız durumlarda sızma testleri yapılmış ve elde edilen sonuçlar değerlendirilmiştir. Web Uygulama Güvenlik Duvarı (WAF) kullanıldığında, web uygulaması güvenlik zafiyetlerinin büyük bir kısmının engellendiği gözlemlenmiştir. Çalışma kapsamında en başarılı saldırı yöntemi Dosya Dahili Etme (File Inclusion) olmuştur. En başarısız saldırılar ise Brute Force ve Zayıf Oturum ID (Weak Session ID) saldırılarıdır. Daha iyi sonuçlar elde etmek için güvenlik duvarlarında kullanılan kuralların kapsamının genişletilmesi ve başarı oranının artırılması önerilmektedir. Ayrıca, güvenlik duvarları kullanılarak farklı güvenlik yapılandırmaları oluşturmak da mümkündür.

5.TEŞEKKÜR

Çalışma sürecinde bana yol gösteren ve fikirleriyle destek olan değerli Ebu Yusuf Güven hocama teşekkür ediyorum.

6.KAYNAKÇA

[1]Saygılı E.,Kılıç F.(2019).Web Uygulama Güvenliği: Bir Örnek Çalışma Web Application Security: A Case Study ,45-48

[2]ALAGÖZ,M.(2020).Açık Kaynak Web Uygulama Güvenlik Duvarı ModSecurity'nin Kullanılabilirlik Analizi ,35-36

[3] Wool, A. (2010). Trends in firewall configuration errors: Measuring the holes in swiss cheese. IEEE Internet Computing, 14(4), 58-65.

[4]Raja F., Hawkey K., and Beznosov K., (2009) "Revealing hidden context: Improving mental models of personal firewall users," SOUPS 2009 - Proc. 5th Symp. Usable Priv. Secur., 2009, doi: 10.1145/1572532.1572534.

[5]Wool A., (2004). "A Quantitative Study of Firewall Configuration Errors," Computer (Long. Beach. Calif)., vol. 37, no. 6, pp. 62–67, doi: 10.1109/MC.2004.2.

[6]Voronkov A., Iwaya L. H., Martucci L. A., and Lindskog S.,(2017). "Systematic literature review on usability of firewall configuration," ACM Comput. Surv., vol. 50, no. 6, doi: 10.1145/3130876.

[7]Clincy V. , Shahriar H., (2018). "Web Application Firewall: Network Security Models and Configuration," Proc. - Int. Comput. Softw. Appl. Conf., vol. 1, pp. 835–836, doi: 10.1109/COMPSAC.2018.00144.

[8] CryptoCat (2021). 9 - Weak Session IDs (low/med/high) - Damn Vulnerable Web Application (DVWA) [Video]

https://www.youtube.com/watch?v=xzKEXAdlxPU

[9] Dockerized Mod Security WAF, Zeyad Abdulaban https://medium.com/cloud-native-daily/dockerized-mod-security-waf-c3e7233be002

[10] Top 10 Open Source Firewall https://www.openappsec.io/post/top-10-free-wafs-web-application-firewalls-for-2024

[11]Siber Saldırılar ve Etik Hacking,Bozburun R.(16 Nisan 2020), File Inclusion Açığı

https://sibersaldirilar.com/web-uygulama-guvenligi/fi le-inclusion-rfi-lfi/

[12] CryptoCat (2021). 5 – File Upload(low/med/high) - Damn Vulnerable Web Application (DVWA) [Video]

https://www.youtube.com/watch?v=K7XBQWAZdZ4&t=857s

[13] CryptoCat (2021). 1 – Brute Force (low/med/high) - Damn Vulnerable Web Application (DVWA) [Video]

https://www.youtube.com/watch?v=SWzxoK6DAE4

[14] Kamil Gierach-Pacanek ,Run Damn Vulnerable Web Application (DVWA) from a Docker container

https://blog.cyberethical.me/run-dvwa-from-docker