```
#####################################################################
# ModSecurity Reference Manual      http://www.mutasyon.net         #
#####################################################################
# License: Relased under the GPL License                            #
#                        http://www.gnu.org/licenses/gpl.html       #
# Original author:  Breach Security            www.breach.com       #
# Document author:  Bunyamin DEMIR    bunyamindemir~gmail.com       #
#                                           blog.bunyamin.org       #
# Version:          v1.0                                            #
# Created:          30.01.2007                                      #
#####################################################################
```

Mod Security Kullanım Kılavuzu

Mod Security

Not: *Bu döküman "Mod Security Reference Manual" dökümanından yararlanılarak yazılmıştır. Zaman zaman içinde kendi yorumlarım ve örneklerim bulunmaktadır.*

Giriş

Mod Security Nedir?

Mod Security, Web uygulamaları için geliştirilmiş açık kaynak kodlu güvenlik duvarıdır "Web Application Firewall (WAF)". Mod Security web sunucusuna gömülü şekilde çalışır. Kullandığınız ve ya yazdığınız web uygulamaları için saldırı tespit ve engelleme görevini üstlenir.

Neden Kullanırız? Yararları Nelerdir?

1. Mod Seucirty HTTP trafiğini son derece detaylı dinler (Bunu ileride auditlog kavramında göreceğiz). Apachenin loglarını göz önünde tutarsak istek içeriği ve cevap içeriği gibi ibarelerin loglanmadığını görürüz. Oysa Mod Security HTTP trafiği üzerinde her türlü veriyi kayıt altına alma yeteneğine sahiptir. Hatta bu logları gruplamanıza ve ya daha okunur şekilde yazdırmanıza yardımcı olur. Bir çok log analiz standartını desteklemektir. Özellikle kendi içinde guarding log sitilinide bulundurmaktadır.

2. Mod Security`nin bir avantajıda gerçek zamanlı veri analizi yapmasıdır. Bu ne demektir? Kullanıcıların uygulamalarınız üzerinde ve ya web sunucunuza bağlandığı andan itibaren gelen giden veriler üzerinde istediğiniz kontrolleri yapmanız demektir.

3. Saldırı tespit ve önleme için anında müdahaleler yapmanıza yarayan kurallar yazabilirsiniz. Mod Security web uygulamalarınıza erişmek isteyen saldırılara karşı anında tepki verir. Bunu çoğunlukla üç yolla yapar.
    a. **Negatif Güvenlik Modeli (Negative Security Model):** Anormal istekleri, sıra dışı hareketleri ve genel web uygulama ataklarını izler. Kısacası bir çok detaya bakıp (ip adress, oturum, kullanıcı hesabı) bunların sonucuna göre kuralların işlenmesi sağlanır.

    b. **Pozitif Güvenlik Modeli (Positive Security Model):** Bu modeli uyguladığınızda, sadece geçerli tanımladığınız istekler kabul edilir ve bunun dışındakiler tümüyle reddedilir. Bu yaklaşım ağır ve nadiren güncellenen uygulamalarla çok iyi çalışır.

    c. **Bilinen Zayıflıklar ve Açıklar (Known weaknesses and Vulnerabilities):** Mod Security`nin kural dili sayesinde, dışarıdan gelen saldırılara karşı sunucunuza kurallar yardımı ile birlikte yamalar yapmanızı sağlar. Bu yamalar sunucunun kendi açıklarından ziyade üçüncü parti yazılımlardan kaynaklanan açıklardır. Bu yazılımlardan kaynaklanan açıklar yazılım sahibi tarafından güncellenene kadar Mod Security ile yamalar oluşturabilirsiniz. Yani dışarıdan gelen zararlı istekleri azaltmaya yarar. Web uygulamalarının açıklarını düzeltmek bir çok kurumda haftaları buluyor. Mod Security sayesinde uygulamanın kaynak koduna dokunmadan (çoğu zaman erişmeksizin) dışarıdan kurallar ekleyerek güvenlik yamalarınızı oluşturabilirsiniz.

4. Mod Security kural moturu (SecRuleEngin) çok esnek kurallar yazmamızı sağlar. Bu motor aynı zamanda Mod Security`nin asıl amacını taşır. Aynı zamanda HTTP işlem dataları üzerinde bize özel bir programlama dili sunar. –ki bu dil normal firewall kullanıcıları ve ya web uygulaması gelştirenler için çokta yabancı değildir (Özellikle regular expression – düzenli ifadeler, PCRE kütüphanesi kullanılır). Bir diğer hususta zincir kurallar oluşturabilmenizdir. Bsunu bir nevi if kurallarına benzeterek yapılan kural takımıda diyebiliriz. Buda sizin daha kompleks kurallar yazmanız anlamına gelir.

5. Yukarıda da söylediğimiz gibi Mod Security gömülü bir firewall uygulamasıdır. Bunun anlamıda kurulu olan web sunucunuza istediğiniz zaman ilave edebilir ve devre dışı bırakabilirsiniz. Mod Security`nin bu kullanım şekli ile daha önce var olan networkünüzde herhangi bir değişiklik yapmanıza gerek kalmaz. Ayrıca kural dosyalarının Web sunucunuzla bir bağı olmadığından taşınabilirliği gayet kolaydır. Ayrıca SSL tarfiğinide analiz etme yeteneğine sahiptir. Bunu SSL üstünden geçen veriler çözüldükten hemen sonra hayata geçirir. Bir çok işletim sistemiyle birlikte gayet uyumlu çalışır. Genel kullanıcıları FreeBSD, Linux, Windows, Solaris, OpenBSD, NetBSD, AIX, Mac OS X ve HP-UX.

6. Mod Security apachenin Mod proxy uygulaması ile birliktede çalışabilir.

Lisans

ModSecurity iki lisans altında kullanılabilir. Kullanıcılar, Açık Kaynak Kodlu / Bedava Yazılım ürünü olarak GNU General Public License (http://www.gnu.org/licenses/gpl.html) gerekleri altında yazılımı kullanmayı tercih edebilirler. Alternatif olarak: bireysel ve ya site-boyu üretim için son kullanıcı lisansları, uygulamalar, web sunucuları ve ya güvenlik araçları ile kapalı kaynak dağıtımı için OEM ticari lisansları kullanılabilir. Ticari lisanslar ile alakalı daha fazla bilgi için lütfen Breach Security (http://www.breach.com/) ile bağlantıya geçin.

Şimdi Mod Security konfigurasyon direktiflerine giriş yapalım. Bu belirtilen direktifler Apachenin bir çok direktif alanında geçerlidir. Bundan kastımız nedir? Örneğin Virtual hostlar, Locatation, LocatationMatch, Directory gibi kavramların içinde Mod Security direktiflerini kullanabiliriz. Aynı zamanda kurallarınızı ve ya direktiflerinizi farklı dosyalarda tutup bunları gerekli yerlerde "Include" yardımı ile çağırıp kullanabilirsiniz.

Fazla sözü uzatmadan direktiflerimizi görelim...

Konfigurasyon Direktifleri

SecAction

SecAction etkisi herhangi bir kurala gereksinim duymaksızın etkilerin (actions) çalıştırılmasına yarar.

SecAction etki1,etki2,etki3,...

Şeklinde yazılır. Genelde bir kural zincirinin içinde kullanılır. O kuralda kurala bağlı olmayan bazı etkilerin çalıştırılması gerekebilir, Örneğin setsid, setuid, initcol gibi...

SecAction nolog,deny,status:403

SecArgumentSperator

URL kısmından gelen ayıraç (sperator) karakteridir. Ön tanımlı olarak "&" kullanılır. Fakat bazı uygulamalarda";" de kullanılır. Örneğin index.php?deneme=w**&**isim=ahmet**&**id=2

SecArgumentSperator &

SecAuditEngine

Denetleme ve ya kontrol mekanizması anlamına gelir ve üç parametre alır.

- On – Default log işlemlerini tut.
- Off – Default log işlemlerini tutma.
- RelevantOnly – Uyarılar, hatalar ve dikkate alınacak bilgileri kayıt altına alma seçeneğidir.

SecAuditEngine RelevantOnly

SecAuditLog

Denetleme mekanizmasının log dosyasının tutulduğu yolu gösterir.

Not: Sunucu root olarak çalıştığında bu dosya açık kalır. Diğer kullanıcılara bu dosyaya yazma izni vermeyiniz.

SecAuditLog /var/log/modsec_audit.log

SecAuditLogParts

Denetleme mekanizmamızı üstleneceği rollere göre bazı parametreler veririz. Şimdi bunları inceleyelim.

- A – Denetleme log başlığı (zorunlu)

--a588181d-A--
[05/Jan/2007:15:27:58 +0200] DF95fn8AAAEAAFvGCkcAAAAD 192.168.2.3 26969 192.168.2.2 80

Burada 192.168.2.3 ip sine sahip bilgisayar 26969 portunu kullanarak 192.168.2.2 ip sine sahip sunucunun 80 portuna ulaştığını görüyoruz.

- B – İstek başlığı.

www.denemesitesi.com/test

Şeklinde bir yere girmek istedik. Buna göre çıktılara bakalım. Lütfen SecAuditLogRelevantStatus de not found yani 404 hatalarını tuttuğunuza dikkat ediniz. Yoksa bu bu örnek için aynı çıktıyı alamayabilirsiniz.

--0af68c7a-B--
GET **/test** HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Host: www.denemesitesi.com
Connection: Keep-Alive

Gördüğünüz gibi GET içinde istediğimiz dosyayı gönderdi.

- C – İsteğin içerik kısmı (body)

- E – Cevap içeriğinin tümünü verir (curl çıktısı)

- F – Cevap başlığını görüntüler

--0af68c7a-F--
HTTP/1.1 **404** Not Found
Content-Length: 202
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

- H – Denetim loglarımızda bazı şeyleri görmemize yarar. Nedir bunlar?

--50e7de65-H--
Apache-Error: [file "core.c"] [line 3737] [level 3] Invalid method in request def
Apache-Handler: httpd/unix-directory
Stopwatch: 1167984392113505 3876 (444 458 -)
Producer: ModSecurity v2.0.4 (Apache 2.x)
Server: Apache/2.2.3 (FreeBSD) DAV/2 PHP/5.2.0 with Suhosin-Patch

- I
- Z – Son parametremizde log başlığını sonlandırıracağımız anlamına gelir (zorunlu) .

SecAuditLogParts ABIFHZ

SecAuditLogRelevantStatus

Sunucunun verdiği yanıtlara göre log tutmak için parametreler verebilirsiniz. Ne demek istedik? Örneğin sayfayı görüntülemek istediniz ve karşınıza “404 not found” çıktı. İşte bu tur responseler

için kullanırız. Tabi SecAuditLogRelevantStatus ün aldığı parametreler düzenli ifadelerdir. Örneğin Sadece 404 ler tutulsun dersek ^404$ şeklinde parametrelendirmeliyiz.
Aşşağıdakide hem 4 hem 5 ile başlayanlar demek...

SecAuditLogRelevantStatus ^[45]

SecAuditLogStorageDir

SecAuditLogType Concurrent seçeneği ile kullandığımızda logların tutulacağı klasörü burada belirtiyoruz. Dikkat edilmesi gereken ise bu klasörün apache kullanıcılarına yazma hakkı olması gerekir. Çünkü çalışma anında yeni dosyalar üretilecektir.

SecAuditLogStorageDir /usr/local/www/apache22/modseclog

SecAuditLogType

İki seçeneği vardır.

1. Serial : Eğer SecAuditLogType yi Serial modda kullanırsanz yukarıda tanımladığımız SecAuditLog /var/log/modsec_audit.log dosyasına bütün denetlenen logları yazar. Yani kullanıcıya ve ya zamana göre ayırt etmeyip bunları tek log dosyasında tutar.

2. Concurrent : Kontrol mekanizmamızın logları bu sefer dosya olarak kaydolur. SecAuditLogStorageDir /usr/local/www/apache22/modseclog buna bakacak olursak. Bu bizim Concurrent modda loglarımızın tutulacağı yerdir. Loglar şu şekilde tutuluyor.

    1. Öncelikle bir gün klasoru “20070105”
    2. Sonra hareketlerin oluşumuna göre gün-saatdakika dosyası “20070105-1140”
    3. Daha sonrada “20070105-114154-4@Ukhn8AAAEAAGw3CXgAAAAC” O dakikada olan isteklerin dosyaları..

Aynı zamanda SecAuditLog da tanımlamış olduğumuz log dosyasınada bu yukarıda saydıklarımızı yazar.

Örnek olarak;
www.mutasyon.net 85.105.184.100 - - [05/Jan/2007:11:41:54 +0200] "GET /index.peeccc HTTP/1.1" 404 210 "-" "-" 4@Ukhn8AAAEAAGw3CXgAAAAC "-" /20070105/20070105-1141/20070105-114154-4@Ukhn8AAAEAAGw3CXgAAAAC 0 951 md5:ef9327affbbc74afcb21e84215c55cc1

SecAuditLogType Serial

SecChrootDir

Apache`yi jail olarak çalıştırmaya yarar.

Eğer chroot dizininiz yoksa.

```
sijiero# mkdir /chroot/apache
sijiero# cd /chroot/apache
sijiero# mkdir log && mkdir run
sijiero# cd /log
sijiero# touch httpd-error.log
```

yaptıktan sonra...

mod security configurasyon dosyasına. “SecChrootDir /chroot/apache” satırını ekleyiniz.

```
sijiero# apachetl stop && apachetl start
```

```
[notice] ModSecurity for Apache 2.0.4 configured - Apache/2.2.3 (FreeBSD) DAV
[notice] ModSecurity: chroot successful, path=/chroot/apache
```

satırlarını alıyorsanız tamamdır..

SecCookieFormat

Yapılandırmada kullanılacak cookie formatını belirleriz. İki seçenek vardır.

- 0 – Default olarak kullanılır ve bir çok uygulama bu formatta cookie kullanılır.
- 1 - Başka çeşit bir cookie formatı

SecCookieFormat 0

SecDataDir

IP adress datası, session datası gibi dataları saklayan path dir. Apache kullanıcısının okuma hakkı olması gerekiyor. (initcol, setsid kullanıken SecDataDir kullanılacaktır).

SecDataDir /var/storedfile

SecDebugLog

Debug loglarının tutulacağı path i gösterir.

SecDebugLog /var/log/modsec_debug.log

SecDebugLogLevel

Burada ise tutulacak logların seviyelerini belirliyoruz. Seçenekleri alt alta sıralayalım.

0 – Hiç birşey kayıt altına alma.
1 – Request (istek) hatalarını. Yani tanıdığımız errorleri tutmaya yarar.
2 – Uyarıları tutar.
3 – Dikkat edilmesi gereken notları.
4 – O an yapılan işlemler hakkında bilgi verir.
5 – Yukarıdakiler gibi fakat her işlemin içeriğinide tutar.
9 – Bütün debug işlemlerinin hepsini loglar.

Not: 1-3 arasındakileri apache error.log dosyasında tutulmaktadır. Eğer debug işlemi yapacaksanız. Yani hata ayıklama ve yakalama gibi bir işleme ihtiyaç duyarsanız 5 nolu seçeneği kullanınız. 4-5-9 log dosyasında aşırı kayıt bulundurmanıza sebep olabilir. Bundan dolayı debug işleminiz bikttikten sonra 0 şeklinde yani hiç log tutmasın gibi ayarlayabilirsiniz. Zaten gerekli (1-3) loglarınız apache error log dosyasında tutuluyor.

SecDebugLogLevel 0

SecDefaultAction

Bir kural kullanılırken kendi ön tanımlı etkilerimizi belirlemenize yarar... Örneğin siz bir kural yazdınız

SecRule &REUQEST_HEADERS_NAMES "@le 15"

Bu kuralın yapacağı etkiler şunlardır. Reddeder, auditlog tutar, 403 Forbidden hatası verir v.s v.s İşte bu default etkileri tanımlamak için SecDefaultAction kullanırız.. Ön tanımlı olarak

log, auditlog, deny, status:403, phase:2, t:lowercase, t:replaceNulls, t:compressWhitespace

Bu etkileri (action) içermektedir. Bunu değiştirmek isterseniz

SecDefaultAction action1,action2,action3....

SecDefaultAction log,deny, status:403, phase:2, t:lowercase, t:replaceNulls, t:compressWhitespace

SecGuardianLog

Bu da log dosyalarınızı dansguardian log formatında tutmanızı sağlar. Fakat burda level olarak birşey belirleyemiyorsunuz. Apache error.log da tutulan logları sadece guardian log formatında size sunuyor.

`SecGuardianLog` /var/log/modsec_guardian.log

SecRequestBodyAccess

Modsecurity tarafından ön tanımlı olarak gelen istek gövdelerinin işleneceğini gösterir.

Parametreleri:

- On – İstek gövdelerine erişim ver
- Off – İstek gövdelerine erişim verme.

SecRequestBodyAccess On

SecRequestBodyLimit

Modsecurity`in arabelleğe almasını istediğimiz maksimum istek gövde büyüklüğünü gösterir. Eğer istekler vereceğimiz değeri aşar ise modsecurity tarafından red edilir ve apache "413 istek boyutu çok fazla" şeklinde hata mesajı verir.

SecRequestBodyLimit 131072

Not: 131072 byte 128 kb ye karşılık gelir.

Biraz açalım. Örneğin sayfalarınızda inputboxlar ve ya buna benzer kullanıcının veri girdiği kısımlar var. Bu tür kısımlardan gelecek verinin boyutunu kontrol altına almış oluyoruz.

Örnek bir httpd-error.log çıktısı.

[Wed Jan 10 20:43:03 2007] [error] [client 192.168.2.3] ModSecurity: Request body is larger than the configured limit (13). [hostname "192.168.2.2"] [uri "/blog/comment.php?id=56"] [unique_id "CGXqLH8AAAEAAJtfA4cAAAAA"]

Kullanıcının ekranınada aşşağıdaki şekilde bir yazı çıkar.

Request Entity Too Large
The requested resource
/blog/comment.php

does not allow request data with POST requests, or the amount of data provided in the request exceeds the capacity limit.

SecRequestBodyInMemoryLimit

Bellekte saklanacak istek gövdesinin maksimum kaç boyutta olması gerektiğini belirlemeye yarar.

SecRequestBodyInMemoryLimit 131072

SecResponseBodyLimit

Cevap mesajının limitini belirtir. Eğer belirtilen boyutu aşan bir cevap olursa "500 interval server error" hatası verir. 1 GB maksimum desteği vardır. Default olarak 512 KB kullanılır.

SecResponseBodyLimit 524288

SecResponseBodyMimeType

Burada ise cevap mesajlarının mime tipleri kontrol altına alınır. Dikkat etmenizi istediğim bir husus olarak önce istek (request) daha sonra çıktı (response) işlemleri çalışır. Mime tipleri şu yönden önemlidir. Sunucunuz mod_security çalıştığında yavaşlayabilir. Bunun sebebi ise cevap verilerinin tümü üzerinde kontrol yaptığınızdandır. Bunun için size lazım olan mime tiplerini filtrelemeniz gerekiyor. Kullanım şekli "SecResponseBodyMimeType mime/type" dir.

SecResponseBodyMimeType (null) text/plain text/html

SecResponseBodyMimeTypesClear

Cevapdaki mime tiplerini temizlemeye yarar.

SecResponseBodyMimeTypesClear

SecResponseBodyAccess

Default olarak bu değişken kapalı gelir. cevapları tamponlamaya, analiz etmeye ve ya bunları yapmamaya yarar. İk seçeneği vardır.

- On – Cevaplara erişim sağlamanızı sağlar. Dikkat etmeniz gereken ise Mime tiplerine izin verdiğiniz cevaplar üzerinden erişim sağlayacağınızdır. Yani SecResponseBodyMimeType kısmında tanımlamadığınız mimeye erişim sağlayamazssınız.
- Off – Cevaplara erişim sağlayamazssınız.

SecRule

Mod Security`nin en önemli kısmına gelmiş bulunuyoruz. SecRule özelliği ile gelen giden veriler üzerinde denetleme mekanizmaları oluşturabiliyoruz. Bir nevi firewall a kural eklemek gibi düşünülebilinir. Genel olarak kullanılan formu:

SecRule DEĞİŞKENLER OPERATOR [ETKİLER]

Değişkenler: Bu kısımda SecRule nin kontrol mekanizması için hangi değişkenleri kontrol edeceğini belirtiyoruz. Örneğin, REQUEST_URI (/index.php?id=3) kısmına denir..

Şimdi kural yazmaya başlayalım...

SecRule REQUEST_URI "deneme"

Içinde deneme geçen url kısmını kabul etmeyecektir.

SecRule REQUEST_URI|QUERY_STRING "deneme"

Bu şekilde değişkenleri yan yana yazarakda ortak bir kontrol yapabiliriz.

Operatörler: Örneğin bir kural yazdık REQUEST_URI için ve diyoruz ki "login.php" ile bağlananları bizden uzak tut... SecRule REQUEST_URI "login\.php" deny,status:500
Yazdığımız kuralde "login.\php" kısmı izim operatörümüzdür. Bu kısımda düzenli ifadeleri kullanıp daha esnek kurallar yazabilirsiniz.

Örneğin,
SecRule REQUEST_HEADER "@rx firefox"

Şeklinde bir kural yazdık. Bu kuralda "rx" düzenli ifade operatörüdür. Burda diyoruz ki..

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; tr; rv:1.8.1.1) Gecko/20061204 Firefox/2.0.0.1

Auditlog çıktısından görebilirsiniz.. Bu satırdak eğer firefox kelimesi varsa kural doğrulanır.. Görüldüğü gibi önce operatör sonra operatörün arayacağı veya karşılaştıracağı parametreler. Daha fazla ulandırmadan geçiyorum.

Etkiler (actions): Bu ise kuralımız doğrulandığı anda ModSecurity`nin ne yapması gerektiğini söylüyoruz.. Yukarıdaki örnekte kullanıcının isteğini reddedit 500 nolu http hatasını ekrana verir.

SecRuleInheritance

Yazılım dillerindende bildiğimiz gibi kuralların miras bırakılması anlamına gelir. Yazılan kuralların bir alt dizinde geçerli olup olmayacağına karar veririz. Örneğin Apachenin genelinde kurallarımız işlitor fakat biz bir VirtualHost içinde Mod Security kurallarının işlenmemesini istiyoruz. Bunun için Virtual Host içinde bir üst kısım olan Apache genel yapısındaki kuralların kalıtımını engellememiz gerekir. Ve ya aynı şekilde bir üst kısımdan miras kalacak kuralları kullanmayıp kendi kurallarımızı yazacaksak. Yine kalıtımı kapatıp sonra tekrar bu virtual host için kurallarımızı tanımlamalıyız. SecRuleInheritance iki parametre alır.

- On – Bir üst dizinden kuralların kalıtımına izin ver
- Off – Kalıtıma izin verme

Apache dizini içinde bir kaç kural yazalım.

```
SecRuleEngine On
SecDefaultAction log,pass,phase:2
...
...

<VirtualHost *:80>
ServerName site1.com
ServerAlias www.site.com
SecRuleInheritance Off
SecDefaultAction log,deny,phase1,redirect:http://www.site2.com
</VirtualHost>

<VirtualHost *:80>
ServerName site2.com
ServerAlias www.site2.com
SecRuleInheritance On
SecRule ARGS “attack”
</VirtualHost>
```

SecRuleEngine

Modsecurity Kural motorudur. Üç parametreye sahiptir.

1. On – Kuralları işler
2. Off – Kuralları işlemez
3. DetectionOnly – Kuralları işler fakat engelleme işlemlerini yapmaz.

Not: Off veya DetectionOnly olduğu zamanlarda sadece bir kural için “ctl:ruleEngine=on” ile açabilirsiniz. Detay için “ctl” bakınız...

SecRuleRemoveById

Kurallarınıza verdiğiniz id veya id aralıklarını kaldırmanıza yarar.

SecRuleRemoveByID 5 8 “100-150”

Yukarıda 5,8 ve 100 ile 150 arasındaki kural id`lerini kaldırmış olduk. Dikkat edilecek husus eğer id bir aralık değil ise aralarında boşluk eğer aralık ise çift tırnak ve arasında tire işaretiyle yazmanız. Lütfen etkiler (actions) kısmında “id” etkisine bakınız..

SecRuleRemoveByMsg

Bunlada daha önce yazılmış kuralları devre dışı bırakabiliriz. Kullanım şekli.

SecRuleRemoveByMsg REGEXP

SecServerSignature

Sunucunuzu maskelemenize yarar. Bunun için bir kaç örnek vereyim...

SecServerSignature “Netscape-Enterprise/6.0”
SecServerSignature “Microsoft-IIS/5.0”

SecServerSignature “SuperSunucu/1.0"

Bunun gibi bir maskeleme koyduğumuzda...

dev# telnet 192.168.2.2 80
Trying 192.168.2.2...
Connected to localhost.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 16 Jan 2007 21:11:51 GMT
Server: **SuperSunucu/1.0**
X-Powered-By: PHP/5.2.0
Connection: close
Content-Type: text/html

Connection closed by foreign host.

Şeklinde Head bilgisi alırız...

SecTempDir

Tempory dosyalarının tutulacağı dizini belirtiyor. Vereceğiniz path için apache kullanıcısının yazma hakkı olması gerekir.

SecTmpDir /tmp

SecUploadDir

ModSecurity multipart/form-data kodlaması yani POST ile dosya gönderme ve ya PUT isteklerine cevap verebilir. SecUploadDir ise bu dosyaları saklayacağımız yeri belirtir. Dosyalar burada geçici olarak tutulur. Bu direktifi SecUploadKeepFiles ile kullanırız (aşşağıda).
SecUploadDir /tmp/upload

SecUploadKeepFiles

SecUploadDir ile tanımlanan path`e yüklenen dosyaları tutup tutmayacağımızı kararlaştırırız. Üç tane parametresi vardır.

1. On – Dosyaları tutar.
2. Off – Dosyaları tutmaz.
3. RelevantOnly – Sadece ilgili olarak tanımlanmış isteklere ait dosyalar tutulur.

SecUploadKeepFiles On

SecWebAppId

Bir sunucuda farklı uygulamalar (virtualhostlar olabilir) kullandığınızı varsayarsak. Birde sesson bilgilerinin /tmp dizinine yazıldığını düşünelim. Aynı anda farklı uygulamaların aynı session_id yi ve ya user_id yi üretmemesi için bu şekilde uygulamaları bir birinden ayıracak bir başlık eklenmiştir. Örneğin elimde iki tane virtualhost var ve ben bu iki virtualhost arasında bir session çakışmasını engellemek istiyorum

<VirtualHost *:80>
ServerName app1.com
ServerAlias www.app1.com
SecWebAppId "Uygulama1"
SecRule REQUEST_COOKIES:PHPSESSID !^$ chain,nolog,pass
SecAction setsid:%{REQUEST_COOKIES.PHPSESSID}
</VirtualHost>

<VirtualHost *:80>
ServerName app2.com
ServerAlias www.app2.com
SecWebAppId "Uygulama2"
SecRule REQUEST_COOKIES:PHPSESSID !^$ chain,nolog,pass
SecAction setsid:%{REQUEST_COOKIES.PHPSESSID}
</VirtualHost>

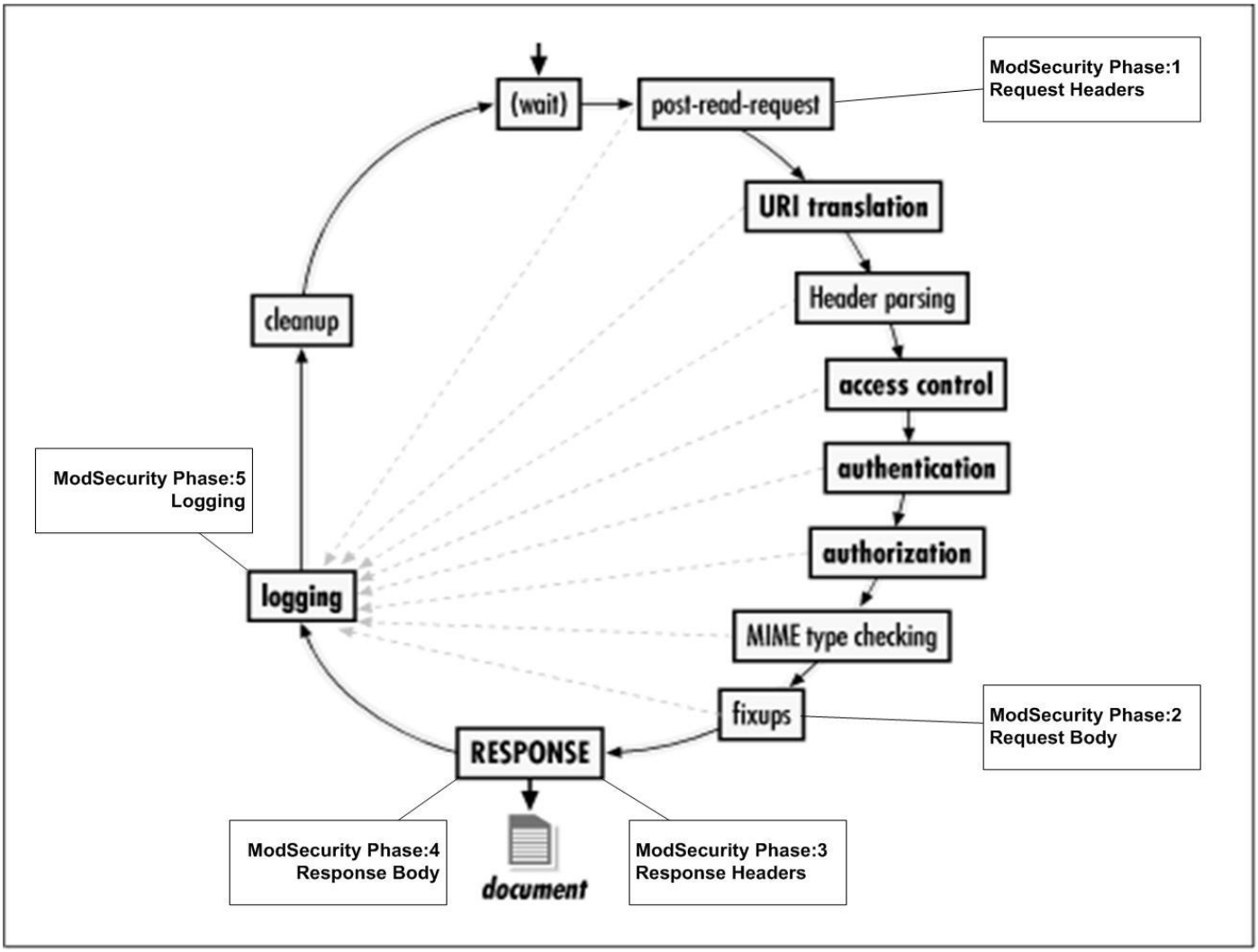
Bu sayede Uygumala1_Session gibi session dosyalarınızı tutarsınız. Bu oturum idlerinin tekilliğini arttırmak için yapılmıştır. WebApp-Info: "Uygulama1" a3510bf7d964fe1be007f27815befe18" "-" (auditlogda H başlığı altında görebilirsiniz).

Bir işlemin safhaları

Mod Security 2.x`de kurallar beş safhada gerçekleşir. Kabaca bir web uygulamasında istek ve buna karşılık gelen bir cevap olduğunu düşünürsek –ki öyle...

1. İstek başlığı (Request headers)
2. İstek içeriği (Request body)
3. Cevap başlığı (Response headers)
4. Cevap içeriği (Response body)
5. Kayıt tutma (Logging)

Altta gördüğünüz diagram ModSecurity`nin işlem safhalarını göstermektedir.



Eğer bir kuralda safha (phase) tanımlanmıyorsa örneğin,

SecRule HTTP_Host “!^$” deny

Bu kural SecDefaultAction ile tanımlanan safhada işler

SecDefaultAction log,deny, status:403, phase:2, t:lowercase, t:replaceNulls, t:compressWhitespace

Demiş isek phasesi tanımlanmayan kural 2 numaralı yani istek içeriği kısmında kontrol edilecektir.

Yada kuralımızı şöyle tanımlarsak;

SecRule HTTP_Host “!^$” “deny,phase:1”

Bu ise 1 numaralı safhada işleme sokulacaktır. Yani kurallarımız safhalara bağlı çalışacaktır.

so even if two rules are adjacent in a configuration file, but are set to execute in different phases, they would not happen one after the other. The order of rules in the configuration file is important only within the rules of each phase. This is especially important when using the "skip" action.

Şimdide yukarıda ayırdığımız safhalardan biraz bahsedelim.

1. İstek başlığı (Request headers)

Bu safhadaki kurallar Apache istek başlıklarını okuduktan hemen sonra devreye girerler. Burada istek içerikleri (request body) okunmaz. Yani istek parametrelerinin hepsi mevcut değildir. Bu kısımda istek başlıklarına göre kurallar yazabilir ve istek içerğinin nasıl olması gerektiğine karar verebilirsiniz.

2. İstek İçeriği (Request body)

Girdi analizinin genel amacı bu safhada gerçekleşir. Uygulamaya yönelik çoğu kural bu kısımda işlenmeli. Genelde kurallarımız cevap üzerinden değilde istek üzerinden olacaktır. Örneğin bir textfield kısmından gelen veriler için yapılacak hamleleri bu kısımda belirleriz. ModSecrutiy istek içeriği safhasında aşşağıdaki kodlama tiplerini destekler.

- Application/x-www-form-urlencoded - veri transferinde kullanılır
- Multipart/form-data – dosya tranferinde kullanılır.
- Text/xml – XML veri geçişinde kullanılır

3. Cevap başlığı (Response headers)

Bu safhada istemcinin girdilerine göre sunucunun istemciye göndereceği cevap başlıkları işlenir.

Run here if you want to observe the response before that happens, and if you want to use the response headers to determine if you want to buffer the response body. Note that some response status codes (such as 404) are handled earlier in the request cycle by Apache and my not be able to be triggered as expected. Additionally, there are some response headers that are added by Apache at a later hook (such as Date, Server and Connection) that we would not be able to trigger on or sanitize. This should work appropirately in a proxy setup however.

4. Cevap içerği (Responset body)

Bu safhanın genel amacı ise cevap analizidir. Cevap içeriği üzerinde yazılacak kurallar bu kısımda yazılır. Örneğin hata mesajlarını, html içeriğini v.s

5. Kayıt tutma (Logging)

Bu safhada kayıt tutma başlamadan önce yazılacak kurallar belirlenir. Örneğin hata mesajlarının nasıl loglanacağı.

Değişkenler

- **ARGS**

  Bununla birlikte tüm argumanları kontrol edebilirsiniz. Bir nevi statik argümandır ve belirlenen statik parametreleri bu argüman ile birlikte kontrol ederiz. Genellikle düzenli ifadeler ile birlikte kullanılır. ModSecurity 1.x de bu değişken QUERY_STRING + POST_PAYLOAD şeklinde idi. Şimdi başlı başına bir değişken olup nerdeyse istek- cevap verilerinin çoğunu denetlemenize yarar.

  Örneğin herhangi bir php sayfamız için bir kural yazalım.

  <LocationMatch "^/bilog/login.php$">
  SecRule ARGS:username "!^\w+$" "deny,log"
  </LocationMatch>

  http://www.site.com/bilog/login.php?username=1 Bu kuralı geçer
  http://www.site.com/bilog/login.php?username=- Bu kurala takılır.

  Yani sayı ve harfler dışında “username” argümanına birşey yollayamazssınız.

- **ARGS_COMBINED_SIZE**

  Argümanların birleşmiş halinin boyutunu kontrol eder.

  SecRule ARGS_COMBINED_SIZE "@gt 9"

  Bu şekilde bir kural eklediniz. @ argumanların sayısını tutar gt operatörü ise büyüklük kontrolü yapar. Şimdi bir örnekle bunu açıklayalım:

  http://www.site.com/page.php?id=5

  şeklinde url adresimiz var. Burada argumanların sayısı (“?id=5“ kısmına bakıyoruz) “id“ ve “5“yani toplamda 3 dür ve bu yukarıda belirlediğimiz size den geçer. Fakat şöyle birşey olmuş olsaydı

  http://www.site.com /page.php?idsinibul=5

  Burada arguman sayımız “idsinibul“ +“5“ sayarsak toplamda 10 dur. Buda yukarıdaki kurala takılır ve Forbidden mesajı alınır. Diğer bir şekildede

  http://www.site.com /page.php?id=54678911

  Buradada “?id=54678911“ kısmındaki arguman sayımız „“id“ +“54678911“ yani 10 dur buda yukarıdaki kurala takılır.

  Buraya kadar parametrenin çalışma mantığından bahsettik. Fakat bukadarla sınırlı değil.

  Örneğin,

```
<form  method="post" action="goster.php">
Name:    <input type="text" name="adi"><br>
Content: <textarea name="yazi" cols="30" rows="10"></textarea><br>
       <input type="submit" value="gonder">
</form>
```

  Şeklinde bir porfumunuz var.. Yukarıdaki kurala göre bu forma sadece “ab” “cd” gibi iki tane karakter yazabilirsiniz. Üçüncü karakteri yazıp gönder dediğinizde 403 hatasını tekrar alacaksınız. Buradan şu sonucu çıkartabiliriz. **ARGS_COMBINED_SIZE** şu sonucu değişkeni POST ve GET ile gönderilen “değişken isimleri + değerler” in toplamını kontrol ediyor. Kullanımı son derece dikkat ister.. Pek kullanışlı gözükmüyor değilmi? Örneğin index sayfanız dahil tüm sisteminizde url kısmında Session ID dönderiyorsunuz. Ozaman “@lt 10” gibi bir parametre ile Session ID uzunluğunu taşmayan herhangi bir url girişine karşı korunmuş olursunuz. Tabi bu “10” uzunluğu şuan keyfidir..

  Dikkat : Özellikle url kısmında session id yi dolaştıran sitelerde bu sorun yaşatabilir onun için eğer bu tarz siteleriniz varsa bu değeri lütfen kontrollü giriniz. URL kısmındaki parametrelerin maksimum olarak kaç olduğunu düşünüp birazda esneklik koyabilirsiniz.

Aynı zamanda POST ile giden verileride sayacağı için bu değerin maximum şekilde tutulması lazım (lütfen dikkat).

- **ARGS_NAMES**

  Değişken ve parametre isimleri için kullanılır. Peki bu parametre nerelere bakar?

  Secrule ARGS_NAMES "password"

  Şek linde bir kural eklediğimizi varsayıyorum. Aşşağıdaki iki örneğede bu kural izin vermez. Onun dışında postla veya getle gönderilen **value**lere yani değerlere izin verir.

  1. http://www.site.com /page.php?password=54678911

  2. <form method="post" action="giris.php">
     Password: <input type="password" name="password">
     <input type="submit" value="gonder"></form>

  Aynı şekilde düzenli ifadeler ile birliktede kontrol edebilirsiniz.

  Secrule ARGS_NAMES "^p.*d$"

  Yani "p" ile başlayıp "d" ile biten değişken isimlerini filtrele...

- **AUTH_TYPE**

  Bir kullanıcı doğrulama işlemindeki metodu tutmaya yarayan değişkendir. Digest ve basic diye parametre alır. Bu özelliği mod proxy ile birlikte kullanmayın.

  SecRule AUTH_TYPE "basic" log,deny,status:403,phase:1

- **ENV**

  Çevresel değişkenleri tutar. Tek parametre ile kullanılır. Aynı zamanda bunun yerine "setenv" de kullanılabilir.. Fakat CGI`da çevresel değişkenleri değiştirmede işe yaramaz..

  SecRule REQUEST_FILENAME "printenv" pass,setenv:tag=suspicious
  SecRule ENV:tag "suspicious"

- **FILES**

  Gerçek dosya isimlerini tutar.

  SecRule FILES "\.conf$" log,deny,phase:2

- **FILES_COMBINED_SIZE**

  Yüklenen dosyaların toplam oyutu. Sadece istek içeriğinde yani ikinci safhada yüklenen dosyalar için geçerlidir.
  SecRule FILES_COMBINED_SIZE "@gt 1000" log,deny,status:403,phase:2

- **FILES_NAMES**
  Dosya yükleme formlarından gelen dosya ismini kontrol eder.

  SecRule FILES_NAMES “^deneme$“ deny,status:403,phase:2

- **FILES_SIZES**
  Yüklenen dosya boyutunu kontrol eder. Buda ikinci fazda gerçerlidir.

  SecRule FILES_SIZES “@gt 100” log,deny,status:403,phase:2

- **FILES_TMPNAMES**

  Tmp dosyalarının isimlerini distkte kontrol etmeye yarar. Genellikle @inspectFile ile kullanılır. İkinci safhada kullanılır (request body).

  SecRule FILES_TMPNAMES “@inspectFile /usr/local/www/apache22/tests/inc_script.pl“

- **HTTP_**
  .
  HTTP_ ile başlayan başlıkların kontrolleri için kullanılır. Örneğin Cookie de sadece PHPSESSID ile başlayan oturumlara izin vermesini istiyoruz.
  (Cookie: PHPSESSID=8ec4a4f7378c3feb5c9f11a8c25762cc)

  SecRule HTTP_COOKIE "^PHPSESSID\$w+$" "deny,log"

- **PATH_INFO**
  Bu parametre scriptlerde kullandığımız path adreslerinin geçişine izin vermemizi sağlar.

  SecRule PATH_INFO “^/(bin|etc|sbin|opt|usr|local|perl)”

- **QUERY_STRING**
  http://www.site.com/index.php?asayi=5&deneme=er göz önünde bulundurursak bunun “asayi=5&deneme=er“ kısmını tutar. Daha genel hal ile site.com/index.php?query_string

  SecRule QUERY_STRING “SELECT”

- **REMOTE_ADDR**
  Ziyaretçinin (client) ip adresi demektir. Örneğin 192.168.5.5 ip sine sahip kullanıcının web sunucuya erişimini istemiyorsunuz.

  SecRule REMOTE_ADDR "^192\.168\.5\.5$"

- **REMOTE_HOST**

  Ziyaretçinin (client) host adresine göre kural eklemenize yarar.

  SecRule REMOTE_HOST “site\.com$”

- **REMOTE_PORT**

  Ziyaretçinin web sunucusuna bağlanırken kullandığı port numarasıdır. Örnek bir kural ekleyelim.

  Mesela root kullanıcı 1024 altındaki portları kullanabilir. Eğer root iseniz size sınırsız geçiş hakkı verecek bir kural ekleyelim.

  SecRule REMOTE_PORT “@lt 1024” pass,setenv:remote_port=privileged

- **REMOTE_USER**

  Ziyaretçinin (client) adını tutar. Eğer bağlantı kurarken (basic ve ya digest) şifre kullanmıyorsanız bu değer boş dönecektir.

  SecRule REMOTE_USER “admin”

  Not: Bu özellik proxy modu kullanırken çalışmaz.

- **REQBODY_PROCESSOR**

  URLENCODED, MULTIPART ve XML işleyiilerini gömme anlamına gelir.

  SecRule REQBODY_PROCESSOR “^XML$” chain
  SecRule XML “@validateDTD /opt/apache-frontend/conf/xml.dtd”

- **REQBODY_PROCESSOR_ERROR**

  0 (hata yok) yada 1 (hata var) Eğer hata alındığında işleyiciyi durdurmak istiyorsanız ikinci safhada bu işlemi kullanmak zorundasınız (phase:2)

  SecRule REQBODY_PROCESSOR_ERROR “@eq 1 “ deny,phase:2

- **REQBODY_PROCESSOR_ERROR_MSG**

  Hata anında mesaj yazdırmak için kullanılır.

  SecRule REQBODY_PROCESSOR_ERROR_MSG “parse hatası” t:lowercase

- **REQUEST_BASENAME**

  Bu değişken ise dosya ismini tutuyor. Örneğin, http:www.site.com/deneme.php adresinde “deneme. .php“ yi tutar.

  SecRule REQUEST_BASENAME “^login\.php$“

- **REQUEST_BODY**

  İstek içeriği kısmında gelen veriler için kural yazmanıza yarar (POST_PAYLOAD verisi). Bu değişkenin geçerli olması için içerik tipi application/x-www-form-urlencoded olmalıdır.

  SecRule REQUEST_BODY “username=\w{25,}\&password=\w{25,}”

- **REQUEST_COOKIES**

  Cookie verilerinin toplamını kontrol etmeye yarar. & (ampersand) işareti toplamda nekadar veri olduğunu hesaplar. Örneğin Cookie içinde hiç bir veri yoksa reddet,

  Secrule &REQUEST_COOKIES “@eq 0” deny,status:403

- **REQUEST_COOKIES_NAMES**

  Istek başlığındaki cookie isminin toplamını kontrol eder.

  SecRule &REQUEST_COOKIES_NAMES:PHPSESSID “@eq 0”

- **REQUEST_FILENAME**

  Bu değişken REQUEST_URI den QUERY_STRING in çıkartılmasıyla elde edilen dosya ismini tutar.

  SecRule REQUEST_FILENAME “^/cgi-bin/login\.php$”

- **REQUEST_HEADERS**

  Bu değişken ile birlikte istek başlıklarının parametrelerini kullanırız. REQUEST_HEADERS:Header-Name şeklinde yazmalısınız. Örneğin tüm istek başlığı parametrelerinin doğruluğuu test edelim (validateUrlEncoding).

  SecRule REQUEST_HEADERS “@validateUrlEncoding”

  Birde sadece “Host” parametresinin konrolüne bakalım

  SecRule REQUEST_HEADERS:Host “^[\d\.]+$” “deny,msg:’Host başlığı sayısal değerli’”

- **REQUEST_HEADERS_ NAMES**

  Bu değişkende İstek başlığındaki tüm parametreleri tutar. Örneğin, Proxy kullanan bir kullanıcının gerçek ip adresi x_forwarded_for da tutulur (gerçi bu değere fazla güvenmeyin). Biz ise proxy kullanan kullanıcıları sistemimizde istemiyoruz.

  SecRule REQUEST_HEADERS_NAMES "^x-forwarded-for" "log,deny,status:403,t:lowercase,msg:'Proxy Kullanıyor "

- **REQUEST_LINE**

  İstek satırı üzerinde işlem yapmanıza yarar. İstek satırı nedir peki?

  dev# telnet 192.168.2.2 80
  Trying 192.168.2.2...
  Connected to localhost.
  Escape character is '^]'.
  HEAD / HTTP/1.0

  Örneğin; istek satırında sadece GET, POST ve HTTP degerleri geçsin.

  SecRule REQUEST_LINE "!(^((?:(?:pos|ge)t))|http/(0\.9|1\.0|1\.1)$)"

- **REQUEST_METHOD**

  İstemcinin kullandığı istek metodunu belirtir. Örneğin; CONNECT ve TRACE dışındaki metodların isteklerine izin vermek istemiyoruz.

  SecRule REQUEST_METOD “^((?:connect|trace))$”

- **REQUEST_PROTOCOL**

  İstek protokolünün versiyon numarasını kontrol etmeye yarar

  SecRule REQUEST_PROTOCOL "!^http/(0\.9|1\.0|1\.1)$"

- **REQUEST_URI**

  URL kısmındaki query parametrelerini filtrelemeye yarar (/index.php?id=3). Örneğin url kısmında “deneme” adlı kelimenin geçmesini istemiyoruz. Bu bizim için sakıncalı bir kelime oldugunu varsayalım ve bu filtrelemeye takılacak senaryoları sıralayalım.

  SecRule REQUEST_URI "deneme"

  http://www.site.com/deneme
  http://www.site.com/deneme.php
  http://www.site.com/index.php?denemeler
  http://www.site.com/index.php?isim=denemeler
  http://www.site.com/index.php?isim=sijiero&deneme=1

  Bunların hepsi bu kurala takılacaktır. Onun için kural olarak vereciniz parametrelere dikkat ediniz. Özallikle düzenli ifadeleri kullanırken.

- **REQUEST_URI_RAW**

  http://www.site.com:80 kısmını tutar. Şimdi bunun içinde bir kural yazalım. Örneğin bir kullanıcının http://www.site.com:22 ve ya https://www.site.com:22 arasine bağlanmasını kısıtlayan kuralı yazalım

  SecRule REQUEST_URI_RAW "^(http|https)\:/.+:22"

- **RESPONSE_BODY**

  Cevap içeriğini kontrol etmek için kullanavilirsiniz. Örneğin ODBC Error hatasının geçtiği sayfalarda deny olmasını istiyoruz..

  SecRule RESPONSE_BODY "ODBC Error"

- **RESPONSE_HEADERS**

  Buda ayrı REQUEST_HEADERS mantığıyla çalışıyor fakat istek değilde cevap başlıkları için kullanılır.

  SecRule RESPONSE_HEADERS:X-cache "MISS"

  bu degisken embebbed modeda iken Server , Date ve Connection başlık bilgilerine ulasamayabilir , cunku bu bilgiler mod-securityinin calismasindan sonraki bir hookta kullaniciya yollanmadan once eklenir.
  proxy modunda direkt client ile araya girdiginden response zaten olusmush oluyor

- **RESPONSE_HEADERS_NAMES**

  Cevap bağlığındaki isimlerini tutar. REQUEST_HEADERS_NAMES ile aynı mantıktadır.

  SecRule RESPONSE_HEADERS_NAMES "Set-Cookie"

- **RESPONSE_PROTOCOL**

  Cevap protokolünün versiyon numarasını tutar.

  SecRule RESPONSE_PROTOCOL "^HTTP\/0\.9"

- **RESPONSE_STATUS**

  Aapache tarafından oluşturulan HTTP cevap durum kodlarını tutar.

  SecRule RESPONSE_STATUS "^[45]"

- **RULE**

  Etkiler (actions) kısmında bahsedilen id, rev, severity ve msg etkilerine erişim sağlamaya yarar.

  SecRule &REQUEST_HEADERS:Host "@eq 0"
  "phase:2,deny,id:1,setvar:tx.varname=%{**rule.id**}"

- **SCRIPT_BASENAME**

  Bu değişken sadece o an çalışan script ismini tutar. Bu değişken mod proxy kullanıldığında geçerli değildir.

  SecRule SCRIPT_BASENAME "^login\.php$"

- **SCRIPT_FILENAME**

  İstek yapılan scriptin sunucudaki fullpath ini tutar. Bu değişkende mod proxy kullanıldığında gerçerli değildir.

  SecRule SCRIPT_FILENAME "^/usr/local/www/apache22/cgi-bin/login\.php$"

- **SCRIPT_GID**

  Kullanılan script in hangi grup tarafından kullanıldığını kontrol ettirir (nümerik değerle). Mod proxy ile çalışmaz.

  SecRule SCRIPT_GID "!^46$"

- **SCRIPT_GROUPNAME**

  Bu ise script`in haklarına sahip grubun adına tutar. Bu değişkende mod prozy ile çalışmaz

  SecRule SCRIPT_GROUPNAME "!^apache$"

- **SCRIPT_MODE**

  Bu değişken dosya izinlerini tutar. (1=çalıştırma, 2=yazama, 4=okuma, 7=okuma/yazma/çalıştırma) Mod proxy ile çalışmaz.

  SecRule SCRIPT_NAME "^(2|3|6|7)$"

- **SCRIPT_UID**

  Buda script sahibinin kullanıcı id`sini verir. Örneğin User id`si 46 olmayanları reddet. (apache user). Bu değişkende mod proxy ile birlikte kullanılmaz.

  SecRule SCRIPT_UID "!^46$"

- **SCRIPT_USERNAME**

  Script sahibinin kullanıcı tutar. Mod proxy ile çalışmaz.

  SecRule SCRIPT_USERNAME "!^apache$"

- **SERVER_ADDR**

  Serverin ip`sini tutar.

  SecRule SERVER_ADDR "^192\.168\.2\.2$"

- **SERVER_NAME**

  Hostname. Bu değişken istemcinin isteği sırasında Host başlığında tutulur.

  SecRule SERVER_NAME “hostname\.com$”

- **SERVER_PORT**

  Web sunucunun dinlediği portu tutar.

  SecRule SERVER_PORT “^80$“

- **SESSION**

  Bilinen session işleminin kontrolü için kullanılır. setsid etkisi çalıştırıldıktan sonra kullanılailir. Aşşağıdaki session işlemini önceleyelim.

  SecRule REQUEST_COOKIES:PHPSESSID !^$ chain,nolog,pass
  SecAction setsid:%{REQUEST_COOKIES.PHPSESSID}
  SecRule REQUEST_URI "^/cgi-bin/finger$" "pass,log,setvar:**session.score**=+10"
  SecRule **SESSION:SCORE** "@gt 50" "pass,log,setvar:**session.blocked**=1"
  SecRule **SESSION:BLOCKED** "@eq 1" "log,deny,status:403"

  İlk satırda PHPSESSID değişkeni varmı diye bakıyor eğer varsa bu kural zinciri başlıyor (chain). İkinci satırda yukarıda da belirttiğimiz gibi setsid etkisi çalıştırılıyor. Üçüncü satırda istemci eğer /cgi-bin/finger/ dizinine girmiş ise (ve ya dosya) session.score=+10 arttırıyor. Dördüncü satırda bu skorun 50 den büyük olup olmadığını sorgulayıp eğer büyükse session.blocked=1 yapıp alttaki satırdada kullanıcıyı sistemden uzaklaştırıyor. Eğer açık gelmediyse etkiler (actions) kısmını okuduktan sonra bir daha örneğe bakınız.

- **SESSIONID**

  Bu değişken setsid ile değiştirilir.

  SecRule SESSIONID !^$ chain,nolog,pass
  SecRule REQUEST_COOKIES:PHPSESSID !^$
  SecAction setsid:%{REQUEST_COOKIES.PHPSESSID}

- **TIME**

  “saat:dakika:saniye” bu formattaki tarihi tutar.

  SecRule TIME "^(([1](8|9))|([2](0|1|2|3))):\d{2}:\d{2}$"

- **TIME_DAY**

  Gerçerli günü tutar (1-31) . Örneğin bir ayın 10. ve 20. gunlei arasında ki bir kuralı tetikleme

  SecRule TIME_DAY “(10|20)”

- **TIME_EPOCH**
  1970 den bu güne kadar geçen zamanı saniye cinsinden tutar.

  SecRule TIME_EPOCH “@gt 10000“

- **TIME_HOUR**
  Saati tutar (0 - 23).

  SecRule **TIME_HOUR** "^(0|1|2|3|4|5|6|[1](8|9)|[2](0|1|2|3))$"

- **TIME_MIN**
  Dakikayı tutar (0 - 59)

  SecRule TIME_MIN "^(3|4|5)"

- **TIME_MON**
  Ayı tutar (0 - 11)

  SecRule TIIME_MON “^1”

- **TIME_SEC**
  Saniyeyi tutar (0 – 59)

  SecRule TIME_SEC “@gt 30”

- **TIME_DAY**
  Günü tutar (0 – 6). Örneğin Hafta sonları çalışacak bir kural ekleyelim.

  SecRule TIME_DAY “^(0|6)$”

- **TIME_YEAR**
  Yılı tutar. (2007)

  SecRule TIME_YEAR “^2007$”

- **TX**
  Bu değişken veri parçalarını saklamaya, anormal bir işlemde skor vermeye v.s için kullanılır. İşlem değişkenleri 1 istek/cevap (request/response) zamanı için ayarlanır. Puantaj ve değerlendirme o an ki istek/cevap sürecini geçemeyecektir.Aşşağıdaki örneği inceleyelim.

  SecRule WEBSERVER_ERROR_LOG "File does not exist"
  "phase:5,pass,**setvar:tx.score=+5**"
  SecRule **TX:SCORE** "@gt 20" deny,log

Ilk satırda eğer “File does not exist” hatası gelirse setvar ile tx.score değerine beş daha ilave ediyoruz. İkinci satırda ise tx:score değişkenini kontrol ediyoruz eğer 20 den büyük olduysa istekleri reddet.

- **USERID**

  Bu değişkenin değeri setuid ile değiştirilir.

  SecAction setuid:%{REMOTE_USER},nolog
  SecRule USERID "Admin"

- **WEBAPPID**

  Bu değişkenin değeri SecWebAppId ile değiştirilir.

  SecWebAppId "WebApp1"
  SecRule WEBAPPID "WebApp1" "chain,log,deny,status:403"
  SecRule REQUEST_HEADERS:Transfer-Encoding "!^$"

- **WEBSERVER_ERROR_LOG**

  Web sunucu tarafından beşinci safhada yani loging kısmında üretilen hataları tutmaya yarar.

  SecRule WEBSERVER_ERROR_LOG "Dosya bulunamadı" "phase:5,setvar:tx.score=+5"

- **XML**

  Standalone olarak kullanabilirsiniz (validateDTD ve valitadeSchema) ve ya bir Xpatch parametresi ile. Örneğin, Xpath parametresi ile kullanımı

  SecRule REQUEST_HEADERS:Content-Type "text/xml"
  phase:1,pass,ctl:requestBodyProcessor=XML,ctl:requestBodyAccess=On
  SecRule XML:/bookstore/book/price/text() "@eq 0" phase:2,deny

Dönüşüm fonksiyonları

Mod Security istek ve cevap bilgilerini aldığında bu verilerin bir kopyasınıda bellekte tutar. Bellekteki bu veriler üzerinde dönüşüm fonksiyonlarını uygular. İşlenmemiş istek/cevap verisi asla değiştirilmez. Dönüşüm fonksiyonlarını her kural için yazmak istemiyorsanız bunu SecDefaultAction parametresiyle belirtmelisiniz. Örneğin her kural için urlDecode, lowercase dönüşümleri uygulansın diyorsanız

SecDefaultAction t:urlDecode,t:lowercase

ModSecurity tarafından desteklenen dönüşüm fonksiyonları:

- **base64Decode**
  Bu fonksiyon base64 ile kodlanmış stringlerin decode eder.

- **base64Encode**
  Bu fonksiyon girilen stringleri base64 ile kodlar.

- **compressWhitespace**
  öntanımlı olarak açık gelmektedir. Bu ise whitespaces karakterlerini (32, \f, \t, \n, \r, \v, 160) bir boşluk karakterine (ASCII 32) ye çevirir ve bir çok boşluk karekterini tek karakter olarak değiştirir.

- **escapeSeqDecode**
  Bu fonksiyon ANSI C escape dizisini decode eder (\a,\b,\f,\n,\r,\t,\v,\\,\?,\',\", \xHH(hexadecimal), \0OOO(octal))

- **hexDecode**
  Hexedecimal formatındaki bir stringi decode eder.

- **hexEncode**
  Bu fonksiyonda girilen stringi hexedecimal olarak kodlar.

- **htmlEntityDecode**
  Bu fonksiyon girdi yapıldığın anda HTML parametrelerini decode eder.
  Mod Security`nin destekledikleri.

  - `&#xHH` ve `&#xHH;`
  - `&#DDD` ve `&#DDD;`
  - `&quot ve &quot;`
  - `  ve  `
  - `&lt ve <`
  - `&gt ve >`

- **lowercase**
  Öntanımlı olarak açık gelir. Tüm karakterleri küçük harf olarak çevirir.

- **md5**
  Girdilerin md5 değerini hesaplar.

- **none**
  Aslında dönüşüm fonksiyonu değildir fakat ModSecurity`de kuralı bağlı tüm dönüşümleri kaldırmaya yarayan bir parametredir. Örneğin,

  SecDefaultAction log,deny,phase:1,t:lowercase,t:removeNulls,t:lowercase

Secrule &REQUEST_COOKIES “@eq 0” log,deny,t:none

Daha önce ön tanımlı olarak belirtilen etkileri kaldırdı.

- **normalisePath**
  Bu fonksiyon ile path tanımlarken kullanılan çoklu slash (///) lar temizlenir.

- **normalisePathWin**
  Buda ters slash işaretini normal slash`a çevirir.

- **removeNulls**
  Girdi içindeki NULL bitlerini siler.

- **removeWhitespace**
  Tüm whitespace (boşluk) karakterlerini siler.

- **replaceComments**
  Bu fonksiyon ise C-stili açıklamaları bir boşluk (ASCII32) ile değiştirir.

- **replaceNulls**
  Öntanımlı olarak açık gelir. Girdilerdeki NULL bitlerini boşluk (ASCII 32) ile değiştirir.

- **urlDecode**
  Bu fonksiyon URL-Encode edilmiş bir girdi dizisini decode eder. Geçersiz kodlanmış diziler (Hexadecimal harici karakter kullanan veya dizinin sonunda bir veya iki karakter eksik olan) çevrilmez. Gerçeksiz ve yanlış kodlamaları tespit etmek için @validateUrlEncoding operatörünü kullanabilirsiniz. Bu fonksiyonun daha onceden url decode edilmiş dizilere sayet tekrardan url decode etmek istemiyorsaniz , kullanilmamasi gerekmektedir.

- **urlDecodeUni**
  ÖurlDecode gibi %xx şeklinde verileri decode eder. Genellikle küçük byte veriler ile kullanılır.

- **urlEncode**
  Bu fonksiyonda URL kodlama stilini kullanarak girdileri kodlar.

- **sha1**
  Girdilerin sha1 değerini hesaplar.

Etkiler (Actions)

Örneğin bir kural eklediniz ve bu kurala uygun bir senaryo gerçekleşti. Varsayıyoyurm ki ben gelen POST verilerinin arasında “deneme“ geçmesin diyorum. “deneme“ geçtiği anda yapacağı işlemleri [ACTIONS] kısmına yazarız. Buna bir kuralın “etki“ kısmıda diyebiliriz. Şimdi bunlardan bahsedip bir çoğuna örnek vereceğiz.

Etkiler kısmına yazılan parametreler beş gruba ayrılır.

1. Yıkıcı etkiler (Disruptive actions): ModSecurity`nin veri üzerindeki en keskin etkileridir. Etkiler: deny,drop,redirect,proxy

2. Yıkıcı olmayan etkiler (Non-disruptive actions): Herhangi bir kuralda kullanılabilir. Etkiler: auditlog , capture, ctl, deprecatevar, exec, expirevar, initcol, log, multiMatch, noauditlog, nolog, sanitiseArg, sanitiseMatched, sanitiseRequestHeader, sanitiseResponseHeader, setuid, setenv, setvar, skip, t, xmlns

3. (Flow actions): Bir kural zincirinin ilk kuralında kullanılır.
Etkiler : allow,chain,pass,skip

4. Yardımcı veri etkileri (Meta-data actions): Bir kural zincirinin ilk kuralında kullanılır. Etkiler: id, msg, rev, severity,phase,loginolohiauditloginoauditlog

5. Veri etkileri (Data actions): capture, status, t, xmlns

- **allow**

  Kural doğrulandığı izin ver anlamına gelir. O an geçerli işlem safhasında çalışır. Tüm kuralları göz ardı ederek bir isteğe izin vermek ise „“ctl“ etkisi ile birlikte kullanmalıyız (Ctl aşşağıda). Örneğin bir sunucuda yöneticisiniz ve sizin ip adresinize her türlü geçiş hakkını vermek istiyorsunuz. IP:192.168.2.2

  SecRule REMOTE_ADDR “^192\.168\.2\.2$” allow,ctl:ruleEngine=Off

- **auditlog**

  Kural doğrulandığında işlemin ayrıntılarını kayıt tutmaya yarar. Aouditlog için yukarıya bakınız... Örneğin şüphelendiğimiz ir ip adresi varve bunun tüm işlemlerini ayrıntılı olarak audit loga yazmasını istiyoruz. Şüphheli Ip adresi: 192.168.2.100

  SecRule REMOTE_ADDR “^192\.168\.2\.100$” auditlog,allow

- **capture**

  Düzenli ifadeler ile birlikte kullandığınızda yapacağı etki, düzenli ifadenin yakaladığı değerin bir kopyasını başka bir dönüşüm içinde kullanmanızı sağlar. 0 ile tüm patern i

alırsınız 1 ilede ilk parantezde olan paterni. Bu şekilde 0-9`a kadar patern tutup yakalayabilirsiniz.
SecRule REQUEST_BODY "^username=(**\w{25,}**)" phase:2,**capture**,t:none,chain
SecRule TX:**1** "(?:(?:a(dmin|nonymous)))"

- **chain**

  Kural zincirlemesidir. Chain i kullandığınız kuralda zincir başlar ve son kurala kadar tanımladığınız ara kuralların hepsinin eşlenmesi yani doğrulanması gerekir ve zincirdeki en son kural zinciri bitirir.. Şimdi bir kural zinciri oluşturalım. Web sunucumda öyle bir alan varki sadece kendi IP`imize erişim izni vermek ve u işlem için log tutmamak istiyorum.

  SecRule REQUEST_URI ”^/test/login\.php$” chain
  SecRule REMOTE_ADDR “^192\.168\.2\.3$” log,auditlog,allow

- **ctl**

  Ctl bize yapılandırma anında kullandığımız direktifleri değiştirme hakkı verir ve ya bir virtualhost içinde auditlog tutulmasını istiyoruz. Fakat diğer yerlerde auditEngine=off değerinde. İşte bu işlemi belli bir kısımda ve ya sadece belli bir kural için ctl ile aktif hale (ve ya pasif) getiriyoruz.

  SecRule REQUEST_CONTENT_TYPE ^text/xml pass,ctl:requestBodyProcessor=XML

  ModSecurity ön tanımlı olarak URLENCODED ve MULTIPART processor lerini kullanır. Eğer XML datası için istek yapacaksak requestBodyProcessor=XML şeklinde çalıştırmamız gerekir. İşte herhangi bir kural için ctl ile bir processor çalıştırdık. Bu diğer kuralları etklilemez.

  Ctl ile kullanabileceğimiz yapılandırma direktifleri:

  1. auditEngine
  2. auditLogParts
  3. debugLogLevel
  4. requestBodyAccess
  5. requestBodyLimit
  6. requestBodyProcessor
  7. responseBodyAccess
  8. responseBodyLimit
  9. ruleEngine

- **deny**

  Kural doğrulandığında işleminizi bitirir ve HTTP 413 Forbidden hata kodunu dönderir. Tabi siz “deny” yaptığınız anda durum kodunu değiştirebilirsiniz (status ile).Örneğin bir kullanıcı ip sini web sunucudan uzak tutmak istiyorsunuz.

  SecRule REMOTE_ADDR “^192\.168\.2\.100$” “auditlog,**deny**,msg:’Zararlı kullanıcı’”

- **deprecatevar**

  Bir sürecin yaşam süresini belirlemeye yarar. Örneğin, her 300 saniyede bir session.core değerini 60 olarak ata. Burada verilen değerin her zaman pozitif olması gerekir.

  SecAction deprecatevar:session.score=60/300

- **drop**

  Bu paramere ile kullanıcıyı reddetmek yerine direk "bağlantıyı koparma" işlemi yani TCP closed connection uygulanır.

  SecAction initcol:ip=%{REMOTE_ADDR},nolog
  SecRule ARGS:login "!^$"
  nolog,phase:1,setvar:ip.auth_attempt=+1,deprecatevar:ip.auth_attempt=20/120
  SecRule IP:AUTH_ATTEMPT "@gt 25" "log,**drop**,phase:1,msg:'Brute Force Attack'"

- **exec**

  Bu parametre ile kural doğrulandığı anda script/binnary kod çalıştırabilirsiniz. Bunun için çalıştırılan script/binary nin tam yolunu vermelisiniz.

  SecRule REQUEST_URI "^/test/login\.pl" "log, exec:/usr/local/www/test.pl".

- **expirevar**

  Adındanda anlaşılacağı gibi bi değişkenin zaman aşımına uğramasını kontrol ediyor (saniye cinsinden).

  SecRule REQUEST_COOKIES:JSESSIONID "!^$" nolog,phase:1,pass,chain
  SecAction setsid:%{REQUEST_COOKIES:JSESSIONID}
  SecRule REQUEST_URI "^/cgi-bin/script\.pl"
  "log,allow,setvar:session.suspicious=1,expirevar:session.suspicious=3600,phase:1"

  Bu örnekte JSESSIONID geldiğinde zincir başlatılıyor ve setsid değişkeniyle atama yapılıyor. Eğer kullanıcı o anda cgi-bin/script.pl dosyasını kullanıyorsa 3600 saniyelik bir expire zamanı atanıyor. (expirevar:session.blocked:3600)

- **id**

  Kural veya zincir kurallara benzersiz (unique) bir id vermeye yarar.

  SecRule &REQUEST_HEADERS:Host "@eq 0" "log,id:60008,severity:2,msg:'Request Missing a Host Header'"

  Verilebilecek kural ID`leri için aşşağıdaki aralıklara dikkat ediniz.

  1. 1 – 999999: Bu aralıktaki id adresleri kendi özel ihtiyaçlarımız içindir. Başkalaıyla paylaşmamız gerekir.

2. 100,000 - 199,999: Bu da modsecurity motorunun kullanımı için ayrılmıştır, belirlenmiş ID`leri olmayan kurallara eşitlemek için.
3. 200,000 - 299,999: Modsecurity.org tarafından yayınlanan kurallar için ayrımıştır.
4. 300,000 – 399,999: Gotroot.com tarafından yayınlanan kurallar için ayrılmıştır.
5. 400,000 ve üzeri: tahsisedilmemiş aralıklardır.

- **initcol**

  Sürekli olarak toplanmasını istediğiniz dataları tutar. Her bir kural için ya datayı belirtilen SecDataDir path`ine yazar yada bellekte tutar. ModSecurity bu işlem için SDBM veritabanını yani DBM dosyalarını kullanır. Tabi dosyaya yazmak ve okumak yerine bellek işlemleri daha hızlı olacaktır. Örneğin bir ip adresini takip etmek istiyoruz.

  SecAction initcol:ip=&{REMOTE_ADDR},setvar=ip.dummy=1,nolog,pass

  Yukarıdaki takip işlemiyle birlikte aşşağıdaki bazı değişkenleride okuyabiliriz.

  1. CREATE_TIME – Takibin başlangıç zamanı.
  2. KEY – initcol işlemimnin değişkeni (yukarıdaki işlem için IP)
  3. LAST_UPDATE_TIME – Son takibin update zamanı.
  4. TIMEOUT –Takip işlemi ne zaman update edileceği.
  5. UPDATE_COUNTER – Takip başladığından değişmesine kadar ne kadar süre geçti.
  6. UPDATE_RATE – Dakikada bir takibi güncellemek.

  Eğer initcol ile bir veri toplama işi yaparsanız (yukarıdaki takip gibi) IP ile birlikte aynı zamanda SESSON için “setsid” ve USER için “setuid” değişkenlerinide kullanabilirsiniz.

- **log**

  Kural doğrulandığında bunu apachenin error.log dosyasına yaz demektir.

  SecAction initcol:ip=%{REMOTE_ADDR},log

- **msg**

  Hata mesajlarında çıkacak bir mesaj yazısı. Mesajın içeriği apache error log ve auditlog dosyalarında gözükecektir. (Hatayı oluşturan kişinin ekranına değil..)

  SecRule REMOTE_ADDR “^192\.168\.2\.100$” “auditlog,**deny**,msg:’Zararlı kullanıcı’”

- **multiMatch**

  Eğer ModSecurity multiMatch kullanıldığı zaman verilen kuralı dönüşümlerden hem önce hemde sonra kontrol eder. Normalde dönüşüm fonksiyonları uygulandıktan sonra değişkenler değerlendirmeye alınır. Fakat multiMatch ile değişken hem dönüşüm fonksiyonları uygulamadan önce hem de uygulandıktan sonra değerlendirmeye alınır.

SecDefaultAction log,deny,phase:1,t:lowercase,t:removeNulls,t:lowercase SecRule ARGS "attack" multiMatch

- **noauditlog**

  Kural doğrulandığı anda auditlog`a yazma demektir.Dikkat edilmesi gereken ise eğer yapılandırma anında "SecAuditEngine On" şeklinde kullandıysanız yapılan işlemler auditlog`a yazılacaktır. Auditlog`a yazılmasını engellemenin en kesin yolu ise. "ctl:auditEngine=Off" şeklindedir. Yani yapılandırma parametresini bu kural için Off say.

  SecRule REQUEST_HEADERS:User-Agent "^m" ctl:auditEngine=Off,deny

- **nolog**

  Kural doğrulandığındaki hataları apache error log dosyasına yazma demektir.

  SecRule REQUEST_HEADERS:User-Agent "^m" nolog,deny

- **pass**

  Kural doğrulandığında işlemi devam ettir ve bir sonraki kurala bak demektir.

  SecRule REQUEST_HEADERS:User-Agent "^m" nolog,pass

- **pause**

  Kural doğrulandığında yürürlükteki proses işlemi belirtilen milisaniye kadar bekletilir. Genellikle Brute Force ataklarını engellemek için kullanılır. Yani gelen isteği belli bir süre bekletip sonra işleme sokmak ard arda taleplerin sayısının uygulama zamanını arttıracaktır. Eğer bu süre uzun olursa browseriniz uyguladığı işlemi sonlandırabilir.

  SecRule REQUEST_HEADERS:User-Agent "^m" pause:5000,pass

- **phase**

  Kuralın ve ya kural zincirinin uygulanacağı safhayı belirtmeye yarar. Eğer bir kuralda "phase:2" gibi bir ibare varsa bu kuralın sadece 2. fazda kontrol edileceği anlamına gelir.

  SecDefaultAction log,deny,phase:1,t:lowercase,t:removeNulls,t:lowercase
  SecRule REQUEST_HEADERS:User-Agent "Test" log,deny,status:403

- **proxy**

  Proxy arkasında bulunan bir webserverin isteklerini durdurur. Bu Etkinin kullanılabilmesi için "mod proxy" yüklü olması gerekir. Aşşağıdaki örnek eğer site.com proxy adresini kullanan bir istek gelirse eşleştirmeye bakar.

  "Intercepts transaction by forwarding request to another web server using the proxy backend."

  SecRule REQUEST_HEADERS:User-Agent "^m" log,proxy:http://www.site.com/

- **redirect**
  Kural doğrulandığında belirtilen url adresine yönlendirilir.

  SecRule REQUEST_HEADERS:User-Agent "^m" nolog,redirect:http://site.com/hata.htm

- **rev**
  Kural revizyonunu belirtmek için kullanılır. Varsayılan olarak “1” kabul edilir. Kuralın her değişikliğinde bu değer bir attırılmalıdır.

  SecRule REQUEST_METHOD "^PUT$" "id:340002,rev:1,severity:2

- **sanitiseArg**
  * (asteriks) leri çevirmeye yarar. Özellikle audirlog içinde görebilirsiniz.

  SecAction nolog,phase:2,sanitiseArg:password

- **sanitiseMatched**
  İstek argümanı, istek başlığı yada cevap başlığındaki asterix li verileri kontrol etmeye yarar.

  SecRule ARGS_NAMES password nolog,pass,sanitiseMatched

- **sanitiseRequestHeader**
  Sanitieses`i bir istek başlığı ismi olarak kullanmak.

  SecAction log,phase:1,sanitiesRequestHeader:Authorization

- **sanitiseResponseHeader**
  cevap ismi olarak kullanmak.

  SecAction log,phase:3,sanitiseResponseHeader:Set-Cookie

- **severity**
  Kural için belli ağırlık katsayıları vardır. Bu ağırlık katsayılarını atamaya yarar.

  SecRule REQUEST_METHOD "^PUT$" "id:340002,rev:1,severity:2

  Alabileceği atama değerleri

  - 0 = EMERGENCY
  - 1 = ALERT
  - 2 = CRITICAL
  - 3 = ERROR
  - 4 = WARNING
  - 5 = NOTICE

- 6 = INFO
- 7 = DEBUG

- **setuid**

  USER collection`ı kullanmamızı sağlar

  SecAction setuid:%{REMOTE_USER}**,**nolog

- **setsid**

  SESSION collection`ı kullanmamızı sağlar.

  SecRule REQUEST_COOKIES:PHPSESSID !^$ chain,nolog,pass
  SecAction setsid:%{REQUEST_COOKIES.PHPSESSID}

  Bu aksiyonun ilk çağırıldığında session dizisi boş olarak yaratılır. (ön tanımlı değişkenler göze alınmadan düşünülmüştür). Daha sonraki çağrılarda dizinin elemanları bellekten getirelecektir (Bu durum için session objeleri) . İlklendirme işlemi sonrası gelen kurallarda SESSIONID parametresi kullanılabilir olacaktır. Bu aksiyon her uygulamanın kendi sessionlarını kullandığını algılamaktadır. Bu şu anda çalışan uygulama idsini kullanarak session namespacesini yaratacaktır.

- **setenv**

  Çevresel değişkenleri oluşturmak, silmek, veya güncellemeye yarar.

Yeni bir değişken oluşturma	setnv:isim=deger
Değeri silme	setenv:!isim

- **setvar**

  Belirtilen kural gurubunda bir değişkeni oluşturmaya, silmeye ve ya değiştirmeye yarar.

Oluşturmak	setvar:tx.skor=10
Silmek	setvar:!tx.skor
Değiştirmek	setvar:tx.skor=+5

- **skip**

  Bir ve ya daha fazla kuralı göz ardı etmenizi sağlar.

  SecRule REQUEST_URI "^/$" "chain,**skip:2**"
  SecRule REMOTE_ADDR "^127\.0\.0\.1$" "chain"
  SecRule REQUEST_HEADERS:User-Agent "^Apache \(internal dummy connection\)$"
  "t:none"
  SecRule &REQUEST_HEADERS:Host "@eq 0"
  "deny,log,status:400,id:960008,severity:4,msg:'Request Missing a Host Header'"
  SecRule &REQUEST_HEADERS:Accept "@eq 0"
  "log,deny,log,status:400,id:960015,msg:'Request Missing an Accept Header'"

- **status**

  Cevap sırasında oluşturulan durum kodlarını kontrol etmenize yarar. Satatus etkisi apachenin herhangi bir dizin ve ya locatation alanında kullanabilirsiniz. Ayrıca farklı hata sayfalarını görüntülemek isterseniz Modsecurity bunları kabul edip işeleyecektir.

  SecDefaultAction log,deny,status:403,phase1

  deny anında 403 durum koduna yönelmesini istiyorsanız.

- **t**

  Bu aslında tam bir etki değil. Fakat dönüşüm fonksiyonlarını taşır. Bunların istekler ve cevaplar üzerinde yapacakları etkileri gösterir. Eğer yazıyı baştan aşşağı takip ettiyseniz bir çok yerde “t:lowercase” gibi ibarelet görmüşsünüzdür.

  SecDefaultAction log,deny,phase:1,t:lowercase,t:removeNulls
  SecRule REQUEST_COOKIES:SESSIONID "47414e81cbbef3cf8366e84eeacba091" log,deny,status:403,**t:**md5

  Birinci satırda DefaultAction olarak log,deny ve ilk fazda yani başlıkların okunduğu kısımda veriler üzerinde küçük harfe dönüştürme ardından NULL bitlerini silme işlemi yapılır. İkinci satırdada az önce t`ye bağlı yapılan dönüşümler (lowercase,removeNulls) Sonucu gelen veriye birde md5 uygulanıp COOKIE`deki session degeri ile eşitliğine bakılır.

- **xmlns**

  Xpath ifadesi ile birlikte kullanılır ve bu ifadeyi isimalanına (namespace) e kaydeder.

  SecRule REQUEST_HEADERS:Content-Type "text/xml" phase:1,pass,ctl:requestBodyProcessor=XML,ctl:requestBodyAccess=On,xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  SecRule XML:/soap:Envelope/soap:Body/q1:getInput/id() "123" phase:2,deny

Operatörler

Operatörler kurallar içinde kullanılırlar. Zaten yukarıda yapılan örnekler içindede bunlar rastladınız. Şimdi bunları tek tek ele alacağım.

- **eq**

  Nümerik olarak eşitmi diye kontrol eder?

  SecRule &REQUEST_HEADERS_NAMES "@eq 15"

- **ge**
  Büyük eşit mi?

  SecRule &REQUEST_HEADERS_NAMES "@ge 15"

- **gt**
  Büyük mü?

  SecRule &REQUEST_HEADERS_NAMES "@gt 15"

- **incpectFile**
  Operatörün her dosya açma isteğine karşılık verilmiş olan bir dış scripti çalıştırır.

  SecRule FILES_TMPNAMES "@inspectFile /opt/apache/bin/inspect_script.pl"

- **le**
  Küçük eşitmi?

  SecRule &REQUEST_HEADERS_NAMES "@le 15"

- **lt**
  Küçükmü?

  SecRule &REQUEST_HEADERS_NAMES "@lt 15"

- **rbl**
  Parametre olarak IPv4 yada hostname kullanabilirsiniz. RBL kontrolü yapar.

  SecRule REMOTE_ADDR “@rbl blog.mutasyon.net”

- **rx**
  Düzenli ifadeler operatördür. Ön tanımlı operatördür. Eğer “@” tanımlı değilse “rx” kabul edilir.

  SecRule REQUEST_HEADERS:User-Agent “@rx fire”

- **validateByteRange**
  Belirttiğiniz byte aralıklarına düşen değerlerde kontrol etmenize yarar. Bu yığıt taşması (stack overflows) saldırılarını önlemekte faydalı olabilir (çünkü bu saldırılar genelde rasgele ikili -binary- içerik içerirler). Varsayılı aralık değerleri 0 ve 255’tir. Yani bütün bayt değerleri kabul edilir. Bu operatör multipart/form-data (dosya yükleme) kullanıldığında bir POST verisindeki kodlamayı kontrol etmez.

SecRule ARG:text "@validateByteRange 10, 13, 32-126"

- **validateDTD**

  İstek içeriğini XML gibi işlenmesini sağlatan operatördür.

  SecDefaultAction log,deny,status:403,phase:2
  SecRule REQUEST_HEADERS:Content-Type ^text/xml$ phase:1,t:lowercase,nolog,
  pass,ctl:requestBodyProcessor=XML
  SecRule REQBODY_PROCESSOR "!^XML$" nolog,pass,skip:1
  SecRule XML "@validateDTD /path/to/apache2/conf/xml.dtd"

- **validateSchema**

  İstek içeriğini XML gibi işlenmesini sağlatan operatördür.

  SecDefaultAction log,deny,status:403,phase:2
  SecRule REQUEST_HEADERS:Content-Type ^text/xml$ phase:1,t:lowercase,nolog,
  pass,ctl:requestBodyProcessor=XML
  SecRule REQBODY_PROCESSOR "!^XML$" nolog,pass,skip:1
  SecRule XML "@validateSchema /path/to/apache2/conf/xml.xsd"

- **validateUrlEncoding**

  Kullanılan değişkenin Url kodlamasına uygun olup olmadığına bakar.

  SecRule ARGS "@validateUrlEncoding"

- **validateUtf8Encoding**

  Buda UTF-8 ile kodlanmış stringlerin doğruluğuna bakar. Bu operatör multipart/form-data (dosya yükleme) kullanıldığında bir POST verisindeki kodlamayı kontrol etmez. Gerekmez çünkü bu kodlamada URL kodlama kullanılmaz.

  SecRule ARGS "@validateUtf8Encoding"

  UTF-8 bir çok web sunucusunda kullanılmaktadır. Üç tip hatayı kontrol eder:

- Yetersiz bayt. UTF-8 iki, üç, dört, beş ve altı baytlık kodlamaları destekler. ModSecurity bir ve ya daha fazla baytın eksik olduğu durumları bulur.

- Geçersiz kodlama. Bir çok karakterin ilk iki bitlerinin 0x80 olması gerekmektedir. Saldırganlar bunu kullanarak evrensel kod çözücülerini alt etmek için kullanabilirler.

- Çok uzun karakterler. ASCII karakterleri evrensel kod uzayına direk olarak eşlenirler ve bu nedenle sadece 1 bayt ile gösterilirler. Ama, bir çok ASCII karakterleri iki, üç, dört, beş ve altı karakterler şeklinde de kodlanabilir ve böylece kod çözücüleri bu karakterlerin farklı karakterler olduğuna inandırabilirler (ve böylece güvenlik kontrollerini geçebilirler).