



**SİBER AKADEMİ  
EĞİTİM MERKEZİ**

**İSTANBUL GELİŞİM ÜNİVERSİTESİ**

**İGÜ SİBER AKADEMİ**

## **IOT Cihazlarını WAF ile Koruma**

**Hazırlayan**

**Hasan Meriç YILDIZ**

**Ödev Danışmanı**

**Dr. Serkan GÖNEN**

**İSTANBUL – 2024**

# ÖDEV TANITIM FORMU

<b>YAZAR ADI SOYADI</b>	: Hasan Meriç YILDIZ
<b>ÖDEVİN DİLİ</b>	: Türkçe
<b>ÖDEVİN ADI</b>	: IoT Cihazlarını WAF ile Koruma
<b>BÖLÜM</b>	: Ağ Sistemleri
<b>ÖDEVİN TÜRÜ</b>	: Final
<b>ÖDEVİN TES. TARİHİ</b>	: 20.12.2024
<b>SAYFA SAYISI</b>	: 38
<b>ÖDEV DANIŞMANI</b>	: Dr. Serkan GÖNEN

## **Beyan**

Bu ödevin/projenin hazırlanmasında bilimsel ahlak kurallarına uyulduğu, başkalarının ederlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğu, kullanılan verilerde herhangi tahrifat yapılmadığını, ödevin/projenin herhangi bir kısmının bu üniversite veya başka bir üniversitedeki başka bir ödev/proje olarak sunulmadığını beyan eder, aksi durumda karşılaşacağım cezai ve/veya hukuki durumu kabul eder; ayrıca üniversitenin ilgili yasa, yönerge ve metinlerini okuduğumu beyan ederim.

Tarih  
20.12.2024

Öğrenci Adı Soyadı

**Hasan Meriç Yıldız**

**Kabul ve Onay Sayfası**

numaralı Hasan Meriç Yıldız'ın “IoT Cihazları WAF İle Koruma” adlı çalışması, benim tarafımdan Final Projesi olarak kabul edilmiştir.

Dr. Serkan GÖNEN

## Özet

Bu projede, IoT cihazlarının güvenliğini artırmak amacıyla bir Web Application Firewall (WAF) çözümü geliştirilmiştir. Ubuntu işletim sistemi üzerinde ModSecurity kurulumu gerçekleştirilmiş, uygun kural setleri tanımlanmış ve Nginx web sunucusu ile entegrasyonu sağlanmıştır. Sistem testi için MJPG-Streamer aracı kullanılarak bir kablolu kamera, IP kamera olarak simüle edilmiş ve bu cihaz WAF ile entegre edilmiştir. Proje kapsamında yapılan sızma testleri, WAF'ın IoT cihazlarını olası saldırılara karşı korumadaki etkinliğini ortaya koymuştur.

Ayrıca çalışmada, IoT, Helium Network, ve WAF kavramları ele alınmış ve bu teknolojilerin güvenlik açısından önemi tartışılmıştır. Bu çalışma, IoT cihazlarının siber tehditlere karşı korunması için WAF kullanımının önemini vurgulamakta ve güvenli bir ağ altyapısı oluşturmanın adımlarını sunmaktadır.

# İçindekiler Tablosu

ÖDEV TANITIM FORMU .....	1
Özet .....	4
Giriş .....	6
Literatür Taraması.....	7
Materyal ve Metot .....	8
Çalışmada Önerilen Sistem.....	9
Kısaltmalar .....	10
IoT Nedir ve Neden Güvenliği Önemlidir .....	11
Helium Network Nedir .....	12
WAF (Web Application Firewall) Nedir .....	13
WAF (Web Application Firewall) Neden Önemlidir .....	14
ModSecurity WAF Nedir .....	16
IoT Cihazlarını WAF İle Koruma .....	17
Sonuç ve Değerlendirme .....	36
Kaynakça.....	37

# Giriş

IoT cihazlarının sayısındaki hızlı artış ve siber tehditlerin kendini geliştirerek ilerletmesi bu cihazların güvenliği konusundaki endişeleri de beraberinde getirmektedir. Bu bağlamda, IoT cihazlarının korunması için etkili çözümler geliştirilmesi kritik bir öneme sahiptir. Bu çalışma, IoT cihazlarının güvenliğini artırmaya yönelik bir yaklaşımın üzerinde durmaktadır. İlk olarak IoT ve Helium Network kavramlarına odaklanılmış, ardından WAF ve ModSecurity WAF teknolojileri açıklanmıştır. Çalışmanın devamında, Ubuntu işletim sistemi üzerinde ModSecurity, Nginx ve kamera için MJPG-Streamer aracı kullanılarak IoT cihazlarının korunmasını hedefleyen bir uygulama geliştirilmiştir.

# Literatür Taraması

1-Systematic Literature Review of Internet of Things (IoT) Security

Ocak 2021

[https://www.researchgate.net/publication/358022729\\_Systematic\\_Literature\\_Review\\_of\\_Internet\\_of\\_Things\\_IoT\\_Security](https://www.researchgate.net/publication/358022729_Systematic_Literature_Review_of_Internet_of_Things_IoT_Security)

2-Endüstri 4.0 ve IoT Güvenliği İçin Web Uygulama Güvenlik Duvarları Kullanımı

30.09.2021

<https://ieeexplore.ieee.org/document/10193640>

3-Ev Ağlarında IoT Güvenliği: UPnP Protokolü Üzerine Bir İnceleme

07.09.2023

<https://dergipark.org.tr/tr/download/article-file/2235618>



# Materyal ve Metot

## Materyaller:

Ubuntu İşletim Sistemi: Projede kullanılan ana işletim sistemi olarak Ubuntu 22.04 LTS sürümü tercih edilmiştir.

ModSecurity: Web uygulamaları için açık kaynaklı bir güvenlik modülü olan ModSecurity, WAF olarak kullanılmıştır.

Nginx: Web sunucusu olarak Nginx kullanılmış, ModSecurity ile entegrasyonu sağlanmıştır.

MJPG-Streamer Aracı: Kablolu bir kamerayı IP kamera olarak simüle etmek amacıyla MJPG-Streamer aracı kullanılmıştır.

## Metodoloji:

Sistem Kurulumu: İlk olarak, Ubuntu işletim sistemi üzerine ModSecurity ve Nginx kurulumu gerçekleştirilmiştir. Ardından, ModSecurity için uygun güvenlik kuralları seti oluşturulmuş ve Nginx ile entegrasyonu yapılmıştır.

IoT Cihazı Simülasyonu: Gerçek bir IoT cihazının simülasyonu amacıyla, MJPG-Streamer aracı kullanılarak bir kablolu kamera IP kamera olarak yapılandırılmıştır.

Test ve Entegrasyon: IoT cihazı, WAF sistemi ile entegre edilerek, sistemin güvenlik testleri yapılmıştır. Sızma testleri gerçekleştirilmiş, WAF'ın IoT cihazını potansiyel saldırılara karşı koruma etkinliği ölçülmüştür.

## Sızma Testi Yöntemi:

Çalışmada, IoT cihazlarına yönelik olası tehditlerin tespiti amacıyla sızma testleri gerçekleştirilmiştir. Bu testler, cihazın web sunucusuna ve WAF sistemine yapılan saldırı simülasyonlarıyla yapılmıştır.

# Çalışmada Önerilen Sistem

Ubuntu İşletim Sistemi:

Sistem, açık kaynaklı ve güvenli bir platform olan Ubuntu işletim sistemi üzerine kurulmuştur. Ubuntu, güvenlik güncellemeleri ve stabil yapısıyla tercih edilmiştir.

ModSecurity WAF:

Bu sistemde ModSecurity, IoT cihazlarını korumak için temel WAF bileşeni olarak kullanılmaktadır. ModSecurity, HTTP trafiğini denetleyerek, kötü amaçlı saldırılara karşı cihazları korur. Ayrıca, kullanıcı tarafından tanımlanan güvenlik kurallarıyla özelleştirilebilir ve IoT cihazlarına özgü güvenlik önlemleri sağlanabilir.

Nginx Web Sunucusu:

Nginx, web trafiğini yönlendiren ve ModSecurity ile entegre çalışan bir web sunucusu olarak kullanılmaktadır. Nginx, yüksek performansı ve düşük kaynak tüketimi ile IoT cihazları için ideal bir sunucu seçeneğidir.

MJPG-Streamer Aracı:

IoT cihazlarının simülasyonu için, kablolu bir kamera IP kamera olarak yapılandırılmıştır. Bu, gerçek dünyada bir IoT cihazının güvenliğini test etmenin etkili bir yoludur. MJPG-Streamer aracı, kamera akışlarını IP tabanlı video akışlarına dönüştürür ve bu akış ModSecurity tarafından korunur.

# Kısaltmalar

IoT:	Internet of Things - Nesnelerin İnterneti
LoRaWAN:	Long Range Wide Area Network - Uzun Menzilli Geniş Alan Ağı
WAF:	Web Application Firewall - Web Uygulama Güvenlik Duvarı
WEB:	World Wide Web - Dünya Çapında Ağ
DDOS:	Distributed Denial of Service - Dağıtık Hizmet Engelleme Saldırısı
SQL:	Structured Query Language - Yapılandırılmış Sorgu Dili
XSS:	Cross-Site Scripting - Site Arası Script(Betik) Çalıştırma
AWS:	Amazon Web Services - Amazon Web Servisleri
OWASP:	Open Web Application Security Project - Açık Web Uygulama Güvenliği Projesi
PCI DSS:	Payment Card Industry Data Security Standard - Ödeme Kartı Endüstrisi Veri Güvenliği Standardı
IP:	Internet Protocol - İnternet Protokolü
CRS:	Core Rule Set – Çekirdek (Temel) Kural Seti
USB:	Universal Serial Bus - Evrensel Seri Veri Yolu
UVC:	USB Video Class - USB Video Sınıfı
URL:	Uniform Resource Locator - Birleşik Kaynak Konumlayıcı
CURL:	Client URL - İstemci URL

# IoT Nedir ve Neden Güvenliđi Önemlidir

IoT (Internet of Things), internete bağlanabilme özelliđi olan cihaz ve nesnelerin oluşturduđu bir ađdır. İsmi "Internet of Things" olarak geđe de, bu cihazların internete bağli olması zorunlu deđildir. Herhangi bir ađa bağli olması ve ayrı ayrı bir şekilde adreslenebilir olması, IoT olması için yeterli gereksinimlerdir. Bu cihazlara örnek olarak; kamera sistemleri, buzdolabı, klima, termostat gibi ev aletleri, akıllı saatler ve endüstriyel makine izleme sensörleri gösterilebilir. IoT cihazları çoğaldıkça ve IoT teknolojisi genişledikçe, gizlilik ve güvenlik gibi unsurların önemi gün geçtikçe artmaya başlamıştır. IoT cihazları, birbirine bağli yapıları nedeniyle güvenlik ihlallerine karşı savunmasız durumdadır. Örneđin, 2016 yılında yaşanan Mirai botnet saldırısı, IoT cihazlarındaki zayıflıkları kullanarak büyük bir DDoS saldırısı gerçekleştirmiştir. IoT teknolojisinin yaygınlaşmasıyla birlikte, güvenliđi artırmak ve cihazlar arasında daha verimli bir bağlantı sağlamak amacıyla Helium Network gibi yenilikçi ađ çözümleri de geliştirilmektedir.



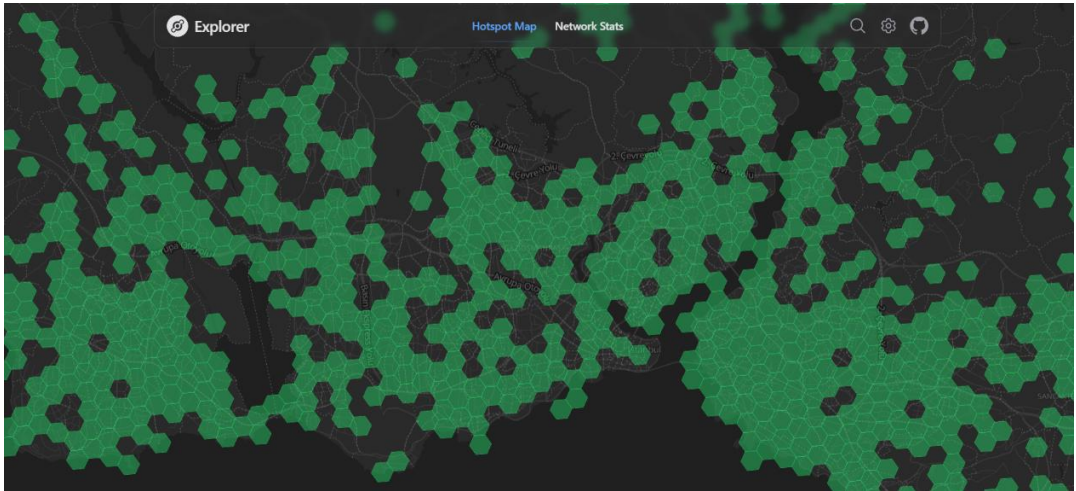
Resim-1 IoT

# Helium Network Nedir

Helium Network, merkezi olmayan ve kripto para birimi Helium Network Token'a (HNT) bağılı bir kablosuz ağıdır. IoT cihazlarının düşük enerji tüketimiyle internete bağlanmasına olanak sağlar.

LoRaWAN (Long Range Wide Area Network) adı verilen, düşük güç tüketimiyle öne çıkan geniş alan ağını kullanan Helium, bölgesel veya küresel ölçekte bir ağ üzerinde kablosuz olarak çalışan ürünlere yönelik haberleşmeyi sağlamaktadır. Bu altyapı, Hotspot'lar aracılığıyla IoT cihazlarının iletişimini mümkün kılmaktadır. Ayrıca Helium, oluşturduğu konsensüs yapısıyla blok zincir dünyasına yenilikçi bir yaklaşım getirmiştir.

Helium ağı üzerinde, nesnelerin birbiriyle iletişimini sağlayan yapılara "Hotspot" denilmektedir. Hotspot kuran kullanıcılar, IoT cihazları için kapsama alanı sunmakta ve bunun karşılığında HNT coin ile ödüllendirilmektedir. Helium üzerinde üç tür Hotspot bulunmaktadır: Full Hotspot, Light Hotspot ve Data Only Hotspot. Her bir Hotspot türü, ağa ilettikleri veri miktarına bağlı olarak HNT ile ödüllendirilir. Helium Hotspot cihazı ile herhangi bir kullanıcı, gerekli tanımlamaları yaptıktan sonra bir Hotspot noktası kurabilir ve sağladığı veriler üzerinden HNT kazanmaya başlayabilir.



**Resim-2 Helium Network Haritası**

Yukarıdaki görselde Helium Network'ün İstanbul'da ne kadar yaygın bir şekilde bulunduğu gösteriliyor.

# WAF (Web Application Firewall) Nedir

WAF (Web Application Firewall), yani web uygulaması güvenlik duvarı, herhangi bir web uygulamasına giden ve uygulamadan gelen tüm HTTP trafiğini yakalar, inceler ve bu trafiğin engellenip engellenmeyeceğine karar vererek işlem sürecini tamamlar. Web uygulaması güvenlik duvarının, geleneksel güvenlik duvarlarından farkı; normal güvenlik duvarları ağı korumak için sunucular arasında güvenlik geçidi görevi görürken, WAF'in doğrudan belirli bir web uygulamasına ait içeriği filtreleyebilmesidir. WAF, HTTP trafiğini inceleyerek SQL enjeksiyonu, XSS (Siteler Arası Betik Çalıştırma) gibi kritik güvenlik zafiyetlerini herhangi bir zarar oluşmadan engelleyebilir.

WAF, kurulum yerine göre üç kategoriye ayrılmaktadır:

## 1. Ağ Tabanlı WAF:

Fiziksel cihazlar üzerinde çalışır. Yüksek hız avantajına sahiptir, ancak maliyetlidir.

Örnekler: Fortinet FortiWeb, Barracuda Networks WAF.

## 2. Host Tabanlı WAF:

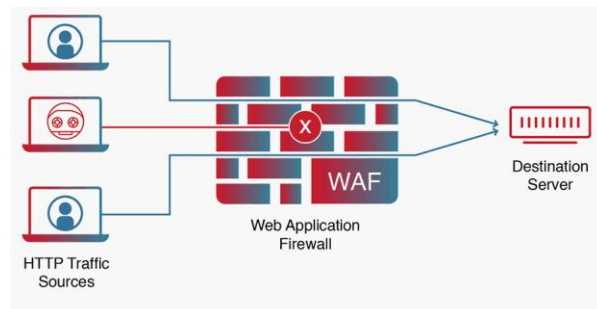
Yazılım olarak çalışır ve sunucunun kaynaklarını kullanır. Bu durum, yapılandırma esnekliği sağlarken performans kayıplarına yol açabilir.

Örnek: ModSecurity.

## 3. Bulut Tabanlı WAF:

Hizmet sağlayıcıları tarafından sunulan bir modeldir. Kurulum ve bakım kolaylığı gibi avantajlar sunar ve genellikle ölçeklenebilir bir yapıya sahiptir.

Örnekler: Cloudflare, AWS WAF.



**Resim-3 Web Uygulama Güvenlik Duvarı (WAF)**

# WAF (Web Application Firewall) Neden Önemlidir

Web Application Firewall (WAF), modern siber tehditlere karşı etkili bir savunma mekanizmasıdır. Bu nedenle, günümüzde birçok firma WAF çözümlerini kullanmakta, henüz kullanmayan firmalar ise bu teknolojiyi benimsemek için çalışmalarını hızlandırmaktadır. WAF'ın başlıca avantajları şu şekildedir:

## **IoT Cihazları İçin Koruma:**

IoT cihazları, genellikle sınırlı işlem kapasitesine ve düşük güvenlik protokollerine sahiptir. Bu durum, IoT cihazlarının büyük veya küçük ölçekli şirketler için ciddi güvenlik riskleri oluşturmalarına neden olur. WAF, bu cihazları hedef alan saldırılara karşı etkili bir koruma sağlar ve bu nedenle büyük önem taşır.

## **Web Uygulamalarına Yapılan Saldırıları Engelleme:**

WAF, web uygulamalarını hedef alan saldırılara karşı kritik bir savunma mekanizmasıdır. Örneğin, SQL enjeksiyon saldırılarıyla bir veri tabanına sızılmaya çalışıldığında, WAF bu saldırıyı engelleyerek veri tabanının güvenliğini sağlar.

## **Gerçek Zamanlı Ağ Trafiği Analizi:**

WAF, ağa gelen tüm trafiği yakalar ve analiz eder. Şüpheli bir trafik tespit edildiğinde devreye girerek ağın zarar görmesini önler.

## **OWASP Top 10 Tehditlerine Karşı Koruma:**

WAF, OWASP (Open Worldwide Application Security Project) tarafından belirlenen en yaygın ve tehlikeli 10 güvenlik açığına karşı koruma sağlar. Bu, uygulamaların güvenliğini artırır ve zarar görmesini engeller.

## **DDoS Saldırılarına Karşı Koruma:**

WAF, DDoS (Denial-of-Service) saldırılarına karşı trafiği filtreleyerek sunucuların aşırı yüklenmesini önler. Örneğin, sadece Türkiye'de faaliyet gösteren bir şirket, ağ trafiğini yabancı ülkelere kapatarak bu ülkelerden gelebilecek DDoS saldırılarını engelleyebilir.

**Regölasyon Uyumu:**

WAF, PCI DSS (Payment Card Industry Data Security Standard) gibi regölasyonlarla uyumluluk saęlayarak, řirketlerin yasal yükümlölüklerini yerine getirmesine yardımcı olur.

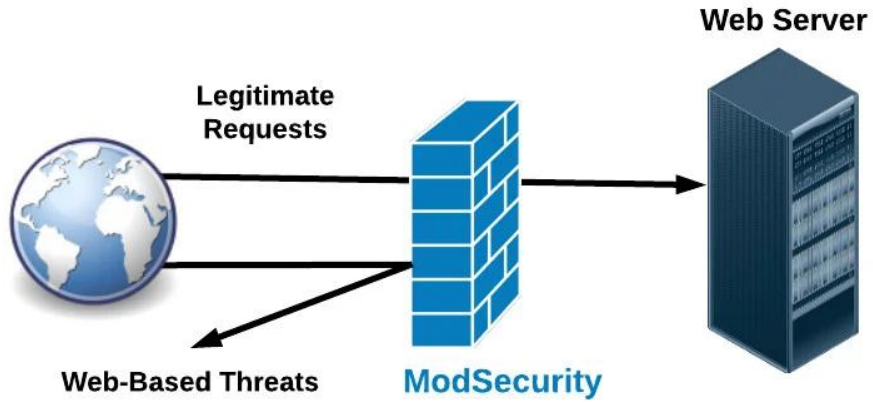


# ModSecurity WAF Nedir

ModSecurity, ModSec olarak da bilinen, açık kaynaklı bir web uygulama güvenlik duvarıdır (WAF). Proje başlangıçta Apache HTTP sunucuları için bir modül olarak tasarlanmış olsa da, zamanla Microsoft IIS ve Nginx desteği eklenerek geniş bir kapsama alanına ulaşmıştır.

ModSecurity, kullanıcı tanımlı kurallara dayalı olarak HTTP iletişimlerini gerçek zamanlı olarak yakalama, izleme ve filtreleme imkanı sunar. Bu işlevsellik, 'SecRules' adı verilen bir kural yapılandırma dili aracılığıyla gerçekleştirilir.

ModSecurity yapılandırılırken genellikle OWASP ModSecurity Çekirdek Kural Seti (OWASP CRS) tercih edilir. Ancak, bu tek kural seti değildir. OWASP CRS, ModSecurity'nin SecRules dilinde yazılmış açık kaynaklı bir kural kümesidir ve web uygulamalarını yaygın güvenlik tehditlerine karşı korumak için kapsamlı bir yapı sunar.



Resim-4 ModSecurity WAF

# IoT Cihazlarını WAF İle Koruma

Bu çalışmada, IoT cihazlarının güvenliğini artırmak amacıyla Ubuntu işletim sistemi üzerinde bir Web Application Firewall (WAF) çözümü geliştirilmiştir. İlk olarak, ModSecurity kurulumu gerçekleştirilmiş ve etkili bir koruma sağlamak için uygun kural setleri tanımlanmıştır. Daha sonra, Nginx web sunucusu kurularak ModSecurity ile entegrasyonu sağlanmıştır. Sistem işlevselliğinin doğrulanması için çeşitli testler uygulanmıştır.

Ayrıca, MJPG-Streamer aracı kullanılarak kablolu bir kamera IP kamera olarak simüle edilmiş ve bu cihaz WAF ile entegre edilmiştir. Bu süreç, IoT cihazlarının siber tehditlere karşı korunmasında WAF kullanımının pratik bir örneğini sunmaktadır. Bu yöntemle, IoT cihazlarına yönelik potansiyel saldırılar engellenmiş ve cihazların güvenliği artırılmıştır.

Dikkat Edilmesi Gerekenler:

ModSecurity'nin kullanılan kural setiyle uyumlu olması büyük önem taşır. Uyumsuz sürümler, sistemin düzgün çalışmasını engelleyebilir ve istenmeyen sorunlara yol açabilir.

**ModSecurity Kurulumu:** Öncelikle aşağıdaki komutları terminalde çalıştırarak sistemimizdeki paketlerin güncel olduğunu doğrulayalım.

```
sudo apt update
sudo apt upgrade
sudo apt install wget apt-transport-https gnupg2 software-
properties-common
sudo apt install g++ flex bison curl apache2-dev doxygen
libyajl-dev ssdeep liblua5.2-dev libgeoip-dev libtool dh-
autoreconf libcurl4-gnutls-dev libxml2 libpcre3 libpcre3-dev
libpcrecpp0v5 libp libxml2-dev git liblmdb-dev libpkgconf3 lmdb-
doc pkgconf zlib1g-dev libssl-dev
```

ModSecurity, Ubuntu 22.04 deposunda mevcuttur. Yine de güncel halini indirmek için aşağıdaki komut terminale yazılır.

```
wget
https://github.com/SpiderLabs/ModSecurity/releases/download/v3.0
.8/modsecurity-v3.0.8.tar.gz
```

İndirdiğimiz dosyayı çıkartalım.

```
tar -xvzf modsecurity-v3.0.8.tar.gz
```

Çıkarttığımız dosyanın dizinine geçmek için aşağıdaki komutları yazıp yapılandırmayı gerçekleştiriyoruz.

Not: make komutu biraz uzun bir süre bekletebilir.

```
cd modsecurity-v3.0.8
./build.sh
./configure
make
make install
```

./build.sh: Projeyi derlemek için kullanılan bir betiği çalıştırır. Genellikle yapılandırma ve bağımlılıkların yüklenmesini başlatır.

./configure: Yazılımın sistemde nasıl yapılandırılacağına dair seçenekleri belirler. Bu adım, derleme için gerekli ortamı hazırlar.

make: Yazılımı derler. Makefile dosyasındaki talimatlara göre kaynak kodunu derler.

make install: Derlenen yazılımı sistemin uygun dizinlerine yükler.

make install komutunun çıktısı aşağıdaki gibidir.

```
make[1]: Entering directory '/home/hasan/modsecurity-v3.0.8'
Making install in benchmark
make[2]: Entering directory '/home/hasan/modsecurity-v3.0.8'
make[3]: Entering directory '/home/hasan/modsecurity-v3.0.8'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/home/hasan/modsecurity-v3.0.8/'
make[2]: Leaving directory '/home/hasan/modsecurity-v3.0.8/'
make[2]: Entering directory '/home/hasan/modsecurity-v3.0.8'
make[3]: Entering directory '/home/hasan/modsecurity-v3.0.8'
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/home/hasan/modsecurity-v3.0.8/'
make[2]: Leaving directory '/home/hasan/modsecurity-v3.0.8/'
make[1]: Leaving directory '/home/hasan/modsecurity-v3.0.8/'
make[1]: Entering directory '/home/hasan/modsecurity-v3.0.8'
make[2]: Entering directory '/home/hasan/modsecurity-v3.0.8'
make[2]: Nothing to be done for 'install-exec-am'.
/usr/bin/mkdir -p '/usr/local/modsecurity/lib/pkgconfig'
/usr/bin/install -c -m 644 modsecurity.pc '/usr/local/mods
make[2]: Leaving directory '/home/hasan/modsecurity-v3.0.8'
make[1]: Leaving directory '/home/hasan/modsecurity-v3.0.8'
```

**Resim-5 make install çıktısı**

**Nginx Sunucusunun Kurulumu:** Şimdi tekrar dizinimizi geri çevirip ModSecurity-Nginx bağlayıcısını GitHub üzerinden indirelim.

```
cd

git clone https://github.com/SpiderLabs/ModSecurity-
nginx.git
```

Sonrasında tekrar aşağıdaki komutu kullanarak Nginx'i resmi kaynağından sistemimize indirelim.

```
wget https://nginx.org/download/nginx-1.20.2.tar.gz
```

ModSecurity'de yaptığımız gibi bunu da çıkaralım.

```
tar xzf nginx-1.20.2.tar.gz
```

Aşağıdaki komutları kullanarak Nginx için bir kullanıcı oluşturalım.

```
useradd -r -M -s /sbin/nologin -d /usr/local/nginx nginx
```

useradd -r -M -s /sbin/nologin -d /usr/local/nginx nginx: nginx kullanıcı hesabı oluşturur. Bu kullanıcı, nginx servisinin çalıştırılması için kullanılır.

```
root@hasan:~# useradd -r -M -s /sbin/nologin -d /usr/local/nginx nginx
```

### Resim-6 Kod parçasığı

Bundan sonra tekrar dizinimizi değiştirelim ve Nginx'in konfigürasyonunu sağlayalım. Ayrıca burada erişim ve hata loglarının tutulması için dizin de belirtiyoruz.

```
cd nginx-1.20.2
./configure --user=nginx --group=nginx --with-pcre-jit --
with-debug --with-compat --with-http_ssl_module --with-
http_realip_module --add-dynamic-module=/root/ModSecurity-
nginx --http-log-path=/var/log/nginx/access.log --error-
log-path=/var/log/nginx/error.log
```

./configure --user=nginx --group=nginx .... : nginx'i yapılandırır ve belirli modülleri, kullanıcı ve grup ayarlarını belirler.

```
root@hasan:~/nginx-1.20.2# ./configure --user=nginx --group=nginx --with-pcre-jit
--with-debug --with-compat --with-http_ssl_module --with-http_realip_module --
add-dynamic-module=/root/ModSecurity-nginx --http-log-path=/var/log/nginx/access
.log --error-log-path=/var/log/nginx/error.log
```

### Resim-7 Konfigürasyon komutu

Tekrar make komutlarını gerçekleştirelim.

```
make
make modules
make install
```

make modules: Nginx için yapılandırılan modülleri derler.

Şimdi aşağıdaki komutla Nginx'in sembolik bağlantısını gerçekleştirelim.

```
ln -s /usr/local/nginx/sbin/nginx /usr/local/sbin/
```

Son olarak aşağıdaki komutu kullanarak Nginx sürümünü doğrulayalım.

```
nginx -V
```

```
root@hasan:~/nginx-1.20.2# nginx -V
nginx version: nginx/1.20.2
built by gcc 13.3.0 (Ubuntu 13.3.0-6ubuntu2~24.04)
built with OpenSSL 3.0.13 30 Jan 2024
TLS SNI support enabled
configure arguments: --user=nginx --group=nginx --with-pcre-jit --with-debug --w
ith-compat --with-http_ssl_module --with-http_realip_module --add-dynamic-module
=/root/ModSecurity-nginx --http-log-path=/var/log/nginx/access.log --error-log-p
ath=/var/log/nginx/error.log
```

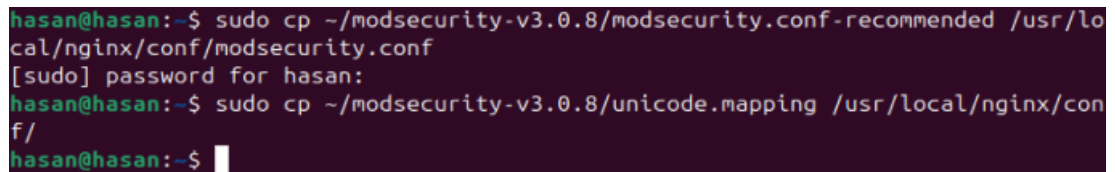
**Resim-8 Nginx versiyon bilgi çıktısı**

**Nginx'i ModSecurity İle Yapılandırma:** Aşağıdaki komutları yeni bir terminalde yazarak yapılandırmamıza başlayalım.

```
sudo cp ~/modsecurity-v3.0.8/modsecurity.conf-recommended
/usr/local/nginx/conf/modsecurity.conf

sudo cp ~/modsecurity-v3.0.8/unicode.mapping
/usr/local/nginx/conf/

sudo cp /usr/local/nginx/conf/nginx.conf{,.bak}
```

A screenshot of a terminal window showing the execution of three commands to install ModSecurity on Nginx. The user 'hasan' is at the prompt. The first command copies the recommended ModSecurity configuration file to the Nginx configuration directory. The second command prompts for a password and then copies the unicode.mapping file. The third command creates a backup of the existing Nginx configuration file.

```
hasan@hasan:~$ sudo cp ~/modsecurity-v3.0.8/modsecurity.conf-recommended /usr/local/nginx/conf/modsecurity.conf
[sudo] password for hasan:
hasan@hasan:~$ sudo cp ~/modsecurity-v3.0.8/unicode.mapping /usr/local/nginx/conf/
hasan@hasan:~$
```

**Resim-9 .conf dosyası ayarlama komutları**

Sonra önceki terminale geçip Nginx konfigürasyon dosyamızı ayarlayalım.

```
nano /usr/local/nginx/conf/nginx.conf
```

Açılan pencerede tüm yazıları silip, sonraki sayfadaki kodları kopyalayıp yapıştıralım.

```
load_module modules/nginx_http_modsecurity_module.so;
user  nginx;
worker_processes  1;
pid                /run/nginx.pid;
events {
    worker_connections  1024;
}
http {
    include          mime.types;
    default_type     application/octet-stream;
    sendfile         on;
    keepalive_timeout  65;
    server {
        listen        80;
        server_name    domain veya ip adresi;
        modsecurity    on;
        modsecurity_rules_file
/usr/local/nginx/conf/modsecurity.conf;
        access_log    /var/log/nginx/access_your-domain.log;
        error_log      /var/log/nginx/error_your-domain.log;
        location / {
            root        html;
            index        index.html index.htm;
        }
        error_page     500 502 503 504    /50x.html;
        location = /50x.html {
            root        html;
        }
    }
}
```

Kaydetmeden önce server\_name yazan yere kendi IP adresimi yazalım. IP numaranızı nasıl öğreneceğinizi bilmiyorsanız sonraki sayfaya bakınız.



IP adresimizi öğrenmek için yeni bir terminalde ip a yazarak IP numaranızı öğrenebilirsiniz. Örneğin bu cihazın IP'si 192.168.1.40

```
hasan@hasan:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:8e:54:cf brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.1.40/24 brd 192.168.1.255 scope global dynamic noprefixroute ens33
        valid_lft 81299sec preferred_lft 81299sec
    inet6 fe80::20c:29ff:fe8e:54cf/64 scope link
        valid_lft forever preferred_lft forever
```

**Resim 10- ip a tablosu**

Server\_name kısmına IP numaramızı yazdıktan sonra CTRL+O, Enter, CTRL+X kısayollarını kullanarak konfigürasyon dosyasını kapatalım.

Şimdi aşağıdaki komutu kullanarak ModSecurity'i etkinleştirelim.0

```
sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/'
/usr/local/nginx/conf/modsecurity.conf
```

**OWASP Kural Setinin Kurulumu:** OWASP'ın en son yayınlanan ModSecurity Çekirdek Kural Setini indirerek başlayalım.

```
cd
git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git /usr/local/nginx/conf/owasp-crs
```

Sonra dosyanın isminde değişiklik yapalım. crs-setup.conf.example'dan crs-setup.conf'a.

```
cp /usr/local/nginx/conf/owasp-crs/crs-setup.conf{.example,}
```

Daha sonra aşağıdaki komutu kullanarak kuralları tanımlayalım.

```
echo -e "Include owasp-crs/crs-setup.conf\nInclude owasp-crs/rules/*.conf" >> /usr/local/nginx/conf/modsecurity.conf
```

**Nginx İçin Systemd Servis Dosyası Kurulumu:** Şimdi Nginx servisini yönetmek için systemd dosyası oluşturalım.

```
nano /etc/systemd/system/nginx.service
```

Açılan dosyaya aşağıdaki satırları ekleyelim.

```
[Unit]
Description=A high performance web server and a reverse proxy server
Documentation=man:nginx(8)
After=network.target nss-lookup.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
ExecStartPre=/usr/local/nginx/sbin/nginx -t -q -g 'daemon on; master_process on;'
ExecStart=/usr/local/nginx/sbin/nginx -g 'daemon on; master_process on;'
ExecReload=/usr/local/nginx/sbin/nginx -g 'daemon on; master_process on;' -s reload
ExecStop=-/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid
TimeoutStopSec=5
KillMode=mixed

[Install]
WantedBy=multi-user.target
```

Daha sonra yine CTRL+O, Enter, CTRL+X yaparak dosyayı kaydedip kapatalım.

Yaptığımız değişiklikleri uygulamak için systemd daemon' u yeniden başlatalım.

```
sudo systemctl daemon-reload  
sudo systemctl start nginx  
sudo systemctl enable nginx
```

sudo systemctl daemon-reload: Sistem servislerini yeniden yükler. Yeni veya değiştirilmiş servis dosyalarını tanıtır.

sudo systemctl start nginx: nginx servisini başlatır.

sudo systemctl enable nginx: nginx servisini, sistem açılışında otomatik olarak başlatılacak şekilde yapılandırır.

Bu komutları uyguladığımızda sistemimiz aktif olarak çalışıyor olacaktır. Bunun denemesini yapmak için. Aşağıdaki komutu çalıştırarak güvenliğimizi test edelim.

```
curl localhost?doc=/bin/ls
```

Bu komut, uzaktan kod çalıştırıp çalıştıramadığımızı anlamamızı sağlar. Eğer sistemimiz sorunsuz bir şekilde kurulduysa aşağıdaki gibi bir çıktı almamız gerekiyor.



```
root@hasan:~# curl localhost?doc=/bin/ls  
<html>  
<head><title>403 Forbidden</title></head>  
<body>  
<center><h1>403 Forbidden</h1></center>  
<hr><center>nginx/1.20.2</center>  
</body>  
</html>
```

**Resim-10 Sızma testi sonucu**

Web uygulama güvenliği duvarı bu komutu saldırı olarak algıladığı için erişim isteğini reddediyor. Ayrıca sisteminizde /var/log/nginx klasörüne gelip error\_your-domain.log dosyasını açtığınızda aşağıdaki gibi bir çıktı verecektir.

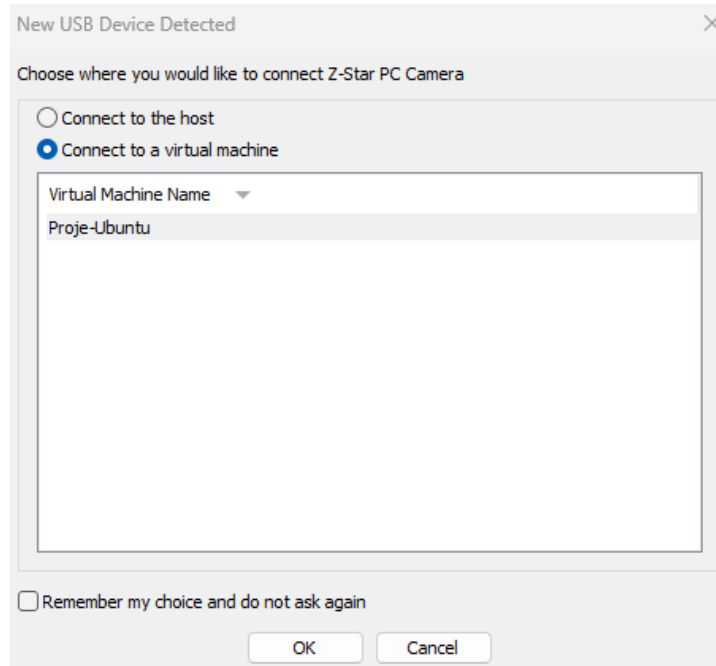
```
2024/12/19 21:40:40 [error] 48793#0: *2 [client 127.0.0.1] ModSecurity: Access denied with code 403 (phase 2). Matched "Operator 'Ge' with parameter '5' against variable 'TX:ANOMALY_SCORE' (Value: '5' ) [file "/usr/local/nginx/conf/owasp-crs/rules/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "80"] [id "949110"] [rev ""] [msg "Inbound Anomaly Score Exceeded (Total Score: 5)"] [data ""] [severity "2"] [ver "OWASP CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [hostname "127.0.0.1"] [uri "/" ] [unique_id "173463364036.564268"] [ref ""], client: 127.0.0.1, server: 192.168.1.40, request: "GET /?doc=/bin/lS HTTP/1.1", host: "localhost"
```

**Resim-11 Erişim reddi log dosyası**

Bu çıktıda ModSecurity'nin çalıştırmaya çalıştığımız /?doc=/bin/lS komutu nedeniyle 403 Forbidden hatası verdiğini çünkü güvenlik kuralı tarafından bir anormallik tespit ettiği ve bu anormalliğin skorunun 5 olduğunu belirtiyor.

**Kamera Modülünün Kurulumu:** Bu projede kablolu bir web kamerası, IoT IP kamera olarak simüle edildiği için MJPG-Streamer aracının kurulumunu gerçekleştirmemiz gerekiyor. Ama önce bilgisayara taktığımız kameranın sistem tarafından tanınıp tanınmadığını kontrol ederek başlamamız gerekiyor.

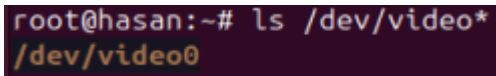
USB Kamera'yı cihaza taktığımızda çıkan pencerede aşağıdaki seçenekleri seçiyoruz.



**Resim-12 USB kamera tanıtma penceresi**

Aşağıdaki komutu kullanarak kameranın bağlı olup olmadığını kontrol edelim.

```
ls /dev/video*
```



```
root@hasan:~# ls /dev/video*  
/dev/video0
```

**Resim-13 Kamera kontrol çıktısı**

Genelde /dev/video0 veya /dev/video1 gibi bir çıktı verecektir. Örneğin bu kullanılan sistemde /dev/video/0 yazıyor.

Kamera akışını test etmek için ffplay aracını kullanarak aşağıdaki kodu yazalım. ffplay toolu yüklü değilse yüklememiz için aşağıdaki komutu çalıştıralım.

```
apt install ffmpeg
```

Kurulumunu gerçekleştirdikten sonra kamerayı test edelim.

```
ffplay /dev/video0
```

Kameramız başarılı olarak çalıştıysa açılır pencerede görüntümüzü göreceğiz. Daha sonrasında açılan pencereyi kapatarak işlemlerimize devam edelim.

Kamerayı yerel ağ üzerinde bir IP Kamerası gibi davranacak şekilde yapılandırmamız için önce kütüphaneleri sonra MJPG-Streamer aracını yükleyelim.

```
sudo apt install build-essential libjpeg-dev cmake -y  
libjpeg-dev -y imagemagick libv4l-dev cmake git
```

Kütüphaneleri yükledikten sonra MJPG-Streamer kaynak kodunu GitHub üzerinden indirelim. Sonra indirdiğimiz dizine geçiş yapalım.

```
git clone https://github.com/jacksonliam/mjpg-streamer.git
cd mjpg-streamer/mjpg-streamer-experimental
```

Dizine geçiş yaptıktan sonra derleyip kurulumunu gerçekleştirelim.

```
make
sudo make install
```

```
-- Installing: /usr/local/share/mjpg-streamer/www/jquery.ui.custom.css
-- Installing: /usr/local/share/mjpg-streamer/www/control.htm
-- Installing: /usr/local/share/mjpg-streamer/www/jquery.js
-- Installing: /usr/local/share/mjpg-streamer/www/JQuerySpinBtn.css
-- Installing: /usr/local/share/mjpg-streamer/www/fix.css
-- Installing: /usr/local/share/mjpg-streamer/www/static_simple.html
-- Installing: /usr/local/share/mjpg-streamer/www/stream.html
-- Installing: /usr/local/share/mjpg-streamer/www/videolan.html
-- Installing: /usr/local/share/mjpg-streamer/www/jquery.ui.tabs.min.js
-- Installing: /usr/local/lib/mjpg-streamer/input_file.so
-- Installing: /usr/local/lib/mjpg-streamer/input_http.so
-- Installing: /usr/local/lib/mjpg-streamer/input_uvc.so
-- Installing: /usr/local/lib/mjpg-streamer/output_file.so
-- Installing: /usr/local/lib/mjpg-streamer/output_http.so
-- Installing: /usr/local/lib/mjpg-streamer/output_rtsp.so
-- Installing: /usr/local/lib/mjpg-streamer/output_udp.so
make[1]: Leaving directory '/root/mjpg-streamer/mjpg-streamer-experimental/_build'
root@hasan:~/mjpg-streamer/mjpg-streamer-experimental#
```

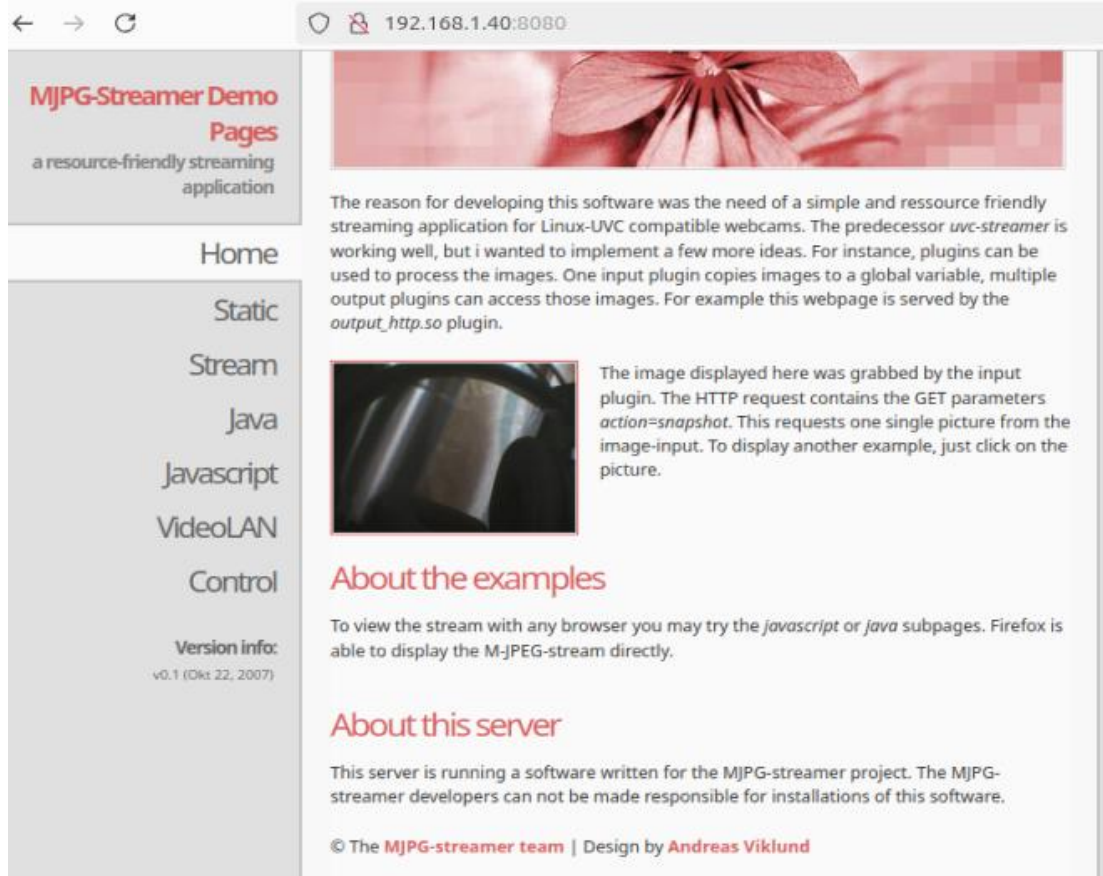
**Resim-14 Sudo make install çıktısı**

Bu komutları da yazdıktan sonra kameranın simüle edilmesini başarmış mıyız diye kontrol edelim.

```
mjpg_streamer -i "input_uvc.so -d /dev/video0 -r 640x480 -f 30" -o "output_http.so -w ./www -p 8080"
```

Portu 8080 olarak belirliyoruz çünkü sistemimizde aktif olarak çalışan Nginx, 80 portu üzerinden çalışmaktadır.

Firefox tarayıcımızı açıp adres çubuğuna `http://ipadresiniz:8080` girdiğinizde MJPEG-Streamer sayfasının congratulations bölümüne gireceksiniz. Bu sayfa bize yapılandırmamızın doğru çalıştığını gösterir.



**Resim-15 MJPEG-Streamer bağlantı sayfası**

Bu sayfayı aldıktan sonra kamerayı çalıştırdığımız terminalde CTRL+C tuş kombinasyonuna basarak yayın akışını durduralım.

Şimdi kameramızın yönlendirme ve proxy işlemlerini `nginx.conf` dosyasından tekrar düzenleyeceğiz.

```
nano /usr/local/nginx/conf/nginx.conf
```

Açılan pencerede tüm metinleri silip sonraki sayfadaki komutları kopyalayıp yapıştıralım.

```

load_module modules/nginx_http_modsecurity_module.so;

user nginx;
worker_processes 1;
pid /run/nginx.pid;

events {
worker_connections 1024;
}

http {
include mime.types;
default_type application/octet-stream;
sendfile on;
keepalive_timeout 65;

# ModSecurity ve Log dosyalarının konfigürasyonu
modsecurity on;
modsecurity_rules_file
/usr/local/nginx/conf/modsecurity.conf;

server {
listen 80;
server_name ipadresiniz;

access_log /var/log/nginx/access_your-domain.log;
error_log /var/log/nginx/error_your-domain.log;

# Kamerayı proxy ile yönlendirme
location /camera/ {
proxy_pass http://127.0.0.1:8080/; # MJPG Streamer portu
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
}

location / {
root html;
index index.html index.htm;
}

error_page 500 502 503 504 /50x.html;
location = /50x.html {
root html;
}
}
}

```



Bu komutları yazdıktan sonra yeni bir terminalde `sudo nginx -t` yazarak yapılandırmayı kontrol edebiliriz.

```
sudo nginx -t
```

```
nginx: the configuration file /etc/nginx/nginx.conf syntax
is ok
nginx: configuration file /etc/nginx/nginx.conf test is
successful
```

Yukarıdaki gibi çıktıyı aldıktan sonra işlemin başarılı olduğunu anlıyoruz.

Sonra `sudo systemctl restart nginx` yaparak Nginx servisini tekrar başlatıyoruz.

```
sudo systemctl restart nginx
```

`sudo systemctl reload nginx` komutu Nginx servisini konfigürasyon değişikliklerini uygulamak için yeniden yükler.

```
sudo systemctl reload nginx
```

Nginx'in doğru bir şekilde çalıştığından emin olmak için `sudo systemctl status nginx` komutunu yazıyoruz.

```
sudo systemctl status nginx
```

```

hasan@hasan:~$ sudo systemctl restart nginx
hasan@hasan:~$ sudo systemctl reload nginx
hasan@hasan:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/etc/systemd/system/nginx.service; enabled; preset: enable>
   Active: active (running) since Thu 2024-12-19 23:03:18 +03; 53s ago
     Docs: man:nginx(8)
  Process: 54839 ExecStartPre=/usr/local/nginx/sbin/nginx -t -q -g daemon on;>
  Process: 54842 ExecStart=/usr/local/nginx/sbin/nginx -g daemon on; master_p>
  Process: 54861 ExecReload=/usr/local/nginx/sbin/nginx -g daemon on; master_>
 Main PID: 54843 (nginx)
    Tasks: 2 (limit: 4558)
   Memory: 44.0M (peak: 53.1M)
      CPU: 498ms
   CGroup: /system.slice/nginx.service
           └─54843 "nginx: master process /usr/local/nginx/sbin/nginx -g daem>
             └─54863 "nginx: worker process"

Dec 19 23:03:17 hasan systemd[1]: Starting nginx.service - A high performance w>
Dec 19 23:03:18 hasan systemd[1]: Started nginx.service - A high performance we>
Dec 19 23:04:01 hasan systemd[1]: Reloading nginx.service - A high performance >
Dec 19 23:04:01 hasan systemd[1]: Reloaded nginx.service - A high performance w>

```

**Resim-16 Nginx mevcut durum kontrolü**

Şimdi kamerayı açtığımız terminale gelip tekrar aşağıdaki komutu çalıştırarak kamerayı aktif hale getiriyoruz

```

mjpg_streamer -i "input_uvc.so -d /dev/video0 -r 640x480 -f
30" -o "output_http.so -w ./www -p 8080"

```

mjpg\_streamer: MJPEG video akışı sunucusunu başlatır. Bu araç, bir USB kameradan video akışını alır ve web üzerinden HTTP ile sunar.

-i: Giriş (input) modülünü belirtir.

"input\_uvc.so": UVC (USB Video Class) destekli bir kameradan video almak için kullanılan giriş modülüdür.

-d /dev/video0: Video kaynağının cihaz yolu. /dev/video0, genellikle ilk USB kamerayı ifade eder.

-r 640x480: Video çözünürlüğü, 640x480 piksel olarak ayarlanır.

-f 30: Video akışının saniyedeki kare sayısını (frame rate) 30 fps olarak ayarlar.

-o: Çıkış (output) modülünü belirtir.

"output\_http.so": HTTP üzerinden video akışını sunmak için kullanılan çıkış modülüdür.

-w ./www: Web dosyalarının bulunduğu dizin. Bu, video akışını izleyebilecek web sayfasını içerir.

-p 8080: HTTP sunucusunun dinleyeceği port numarası. Bu durumda, 8080 portu kullanılır.

Daha sonrasında yine Firefox tarayıcıya girip adres çubuğuna <http://ipadresiniz/camera/> adresine gittiğimizde kameramızı ModSecurity ile başarıyla entegre ettiğimizi görüyoruz.



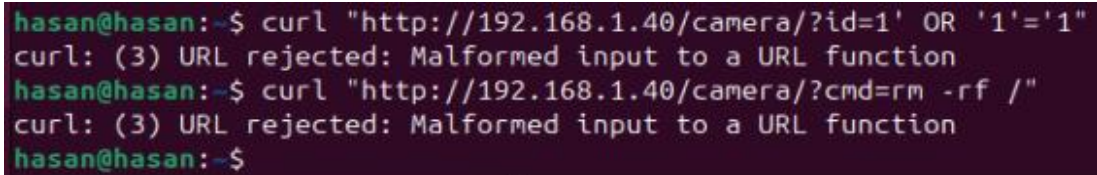
**Resim 17- IP Kamera ve ModSecurity bağlantı testi**

Şimdi son adım olarak kameramız gerçekten WAF ile korunuyor mu onu test edelim. Aşağıdaki saldırı komutlarını terminale yazdığımızda kameramızın WAF tarafından başarıyla korunduğunu göreceğiz.

```
curl http://ipadresiniz/camera/?id=1' OR '1'='1
curl http://ipadresiniz/camera/?cmd=rm -rf /
```

curl http://192.168.1.40/camera/?id=1' OR '1'='1: SQL enjeksiyonu testi yapar, URL parametresine kötü amaçlı veri ekler.

curl http://192.168.1.40/camera/?cmd=rm -rf /: Sistem üzerinde zararlı bir komut çalıştırmayı hedefler (tüm dosyaları siler).



```
hasan@hasan:~$ curl "http://192.168.1.40/camera/?id=1' OR '1'='1"
curl: (3) URL rejected: Malformed input to a URL function
hasan@hasan:~$ curl "http://192.168.1.40/camera/?cmd=rm -rf /"
curl: (3) URL rejected: Malformed input to a URL function
hasan@hasan:~$
```

### **Resim-18 İkinci sızma testi**

Gördüğünüz üzere WAF gelen istekleri reddederek IoT Kameramıza erişmek isteyen kişileri başarıyla durduruyor.

## Sonuç ve Değerlendirme

Bu çalışmada, IoT cihazlarının güvenliğini artırmak amacıyla bir Web Uygulama Güvenlik Duvarı (WAF) çözümü geliştirilmiş ve IoT cihazlarına yönelik potansiyel güvenlik tehditlerine karşı koruma sağlanmıştır. Çalışma kapsamında kullanılan Ubuntu işletim sistemi, ModSecurity WAF, Nginx web sunucusu ve MJPG-Streamer aracı ile IoT cihazlarının güvenliğini sağlamaya yönelik entegre bir sistem geliştirilmiştir.

Proje sonunda gerçekleştirilen sızma testi, WAF'ın IoT cihazlarını dış saldırılara karşı etkin bir şekilde koruduğunu ortaya koymuştur. ModSecurity'nin HTTP trafiğini analiz etmesi ve kötü amaçlı saldırıları tespit ederek engellemesi, sistemin güvenlik seviyesi üzerinde olumlu sonuçlar vermiştir. Bu sayede, IoT cihazlarının ağ üzerinde güvenli bir şekilde çalışması sağlanmış ve cihazların potansiyel tehditlere karşı dayanıklı hale gelmesi hedeflenmiştir.

Bununla birlikte, sistemin etkinliği, sadece temel saldırı türleriyle sınırlı kalmayıp, farklı güvenlik testleriyle genişletilerek daha kapsamlı bir değerlendirmeye tabi tutulabilir. Gerçek dünyadaki IoT cihazlarına entegre edilmeden önce daha geniş kapsamlı saldırı senaryolarıyla test edilmesi, sistemin güvenliğini daha da artırabilir. Ayrıca, ModSecurity'nin sürekli güncellenmesi ve yeni tehditlere karşı uyarlanabilir hale getirilmesi, IoT cihazlarının uzun vadeli güvenliğini sağlayacaktır.

Önerilen sistemin bir diğer avantajı, açık kaynaklı yazılımların kullanılmasıyla düşük maliyetli bir çözüm sunmasıdır. Bu sayede, küçük ölçekli işletmeler ve bireysel kullanıcılar da IoT cihazlarının güvenliğini sağlamak için benzer çözümler geliştirebilirler.

Sonuç olarak, bu çalışma, IoT cihazlarının siber tehditlere karşı korunmasında WAF kullanımının önemini bir kez daha vurgulamaktadır. Gelecekteki çalışmalar, IoT cihazlarının güvenliğini artırmak için daha gelişmiş algoritmalar ve yapay zeka tabanlı çözümlerle desteklenebilir. IoT güvenliği, hızla gelişen bir alan olduğundan, bu tür sistemlerin sürekli olarak izlenmesi ve güncellenmesi gerektiği unutulmamalıdır.

## Kaynakça

1. [https://tr.wikipedia.org/wiki/Nesnelerin\\_interneti](https://tr.wikipedia.org/wiki/Nesnelerin_interneti)
2. [https://en.wikipedia.org/wiki/Internet\\_of\\_things](https://en.wikipedia.org/wiki/Internet_of_things)
3. <https://en.wikipedia.org/wiki/ModSecurity>
4. [https://en.wikipedia.org/wiki/Web\\_application\\_firewall](https://en.wikipedia.org/wiki/Web_application_firewall)
5. [https://en.wikipedia.org/wiki/Helium\\_Network](https://en.wikipedia.org/wiki/Helium_Network)
6. <https://www.helium.com/>
7. <https://explorer.helium.com/>
8. <https://www.bitlo.com/rehber/helium-hnt-nedir>
9. <https://doc.kaas.thalesdigital.io/docs/Features/WAF>
10. <https://owasp.org/www-project-top-ten>