CADDY

Caddy, modern ve açık kaynaklı bir web sunucusudur. En dikkat çeken özelliği, HTTPS sertifikalarını otomatik olarak alması ve yenilemesidir. Bu sayede güvenli web siteleri hızlı ve kolay bir şekilde kurulabilir. Caddy, yapılandırması basit, güvenliği güçlü bir web sunucusu çözümüdür.

UYUMLU OLDUĞU ORTAMLAR:

- LINUX, WINDOWS, MACOS
- DOCKER, KUBERNETES
- VPS, BARE-METAL VE BULUT (AWS, GCP, AZURE)

CORAZA

Coraza, web uygulamaları için açık kaynak kodlu bir WAF (Web Application Firewall) yani Web Uygulama Güvenlik Duvarıdır. Caddy gibi web sunucularına entegre edilerek gelen istekleri denetler. Zararlı istekleri tespit eder ve engeller. Özellikle SQL Injection, XSS gibi saldırılara karşı etkili koruma sağlar.

UYUMLU OLDUĞU SUNUCULAR:

- NGINX (ÖZELLİKLE MODSECURITY DESTEĞİYLE)
- APACHE (DOĞRUDAN DEĞİL, MODSECURITY İLE ENTEGRE)
- CADDY (DOĞRUDAN ENTEGRASYON GELİŞTİRİLİYOR VEYA MÜMKÜN)
- ENVOY PROXY, TRAEFIK GİBİ MODERN PROXY ALTYAPILARI

SAFELINE

SafeLine, web uygulamalarının performansını ve güvenliğini test etmek için kullanılan bir araçtır. Gerçek trafik simülasyonu yaparak sistemin saldırılar karşısındaki davranışını analiz eder. WAF kurulumlarının etkinliğini ölçmek ve zayıf noktaları belirlemek için kullanılır.

UYUMLU OLDUĞU SUNUCULAR / SİSTEMLER:

- GENELLİKLE ÜRETİM/OTOMASYON SİSTEMLERİNE ÖZEL KURULUR
- WINDOWS SERVER (ÇOĞUNLUKLA)
- SQL SERVER VEYA ORACLE GİBİ VERİTABANI SİSTEMLERİYLE ÇALIŞIR
- ON-PREMISE (YEREL AĞDA KURULU) VEYA ÖZEL BULUT ALTYAPILARI

KULLANIM YOĞUNLUĞU VE RAKİP KARŞILAŞTIRMASI

CADDY

Kullanım Oranı:

Caddy, özellikle geliştiriciler ve küçük ölçekli işletmeler arasında popülerdir. Otomatik HTTPS özelliği sayesinde tercih edilmektedir.

- OTOMATİK HTTPS VE TLS SERTİFİKASI YÖNETİMİ SAĞLAR.
- KOLAY YAPILANDIRMA VE KULLANICI DOSTU ARAYÜZ SUNAR.
- GO DİLİYLE YAZILDIĞI İÇİN PERFORMANSI YÜKSEKTİR.

  RAKİPLERİ: NGINX - APACHE HTTP SERVER - LIGHTTPD

CORAZA

Kullanım Oranı:

Coraza, özellikle güvenlik odaklı web uygulamaları geliştiren firmalar ve bireysel geliştiriciler tarafından tercih edilmektedir. Açık kaynaklı olması ve ModSecurity uyumluluğu sayesinde, güvenlik duvarı çözümleri arayanlar arasında popülerdir.

- MODSECURITY KURALLARINI DESTEKLER.
- YÜKSEK PERFORMANSLI VE HAFİF BİR ÇÖZÜMDÜR.
- GO DİLİYLE YAZILDIĞI İÇİN MODERN SİSTEMLERLE UYUMLUDUR.
- AÇIK KAYNAK KODLUDUR, TOPLULUK DESTEĞİ MEVCUTTUR.

  RAKİPLERİ: MODSECURITY - NAXSI - OPENRESTY WAF

SAFELINE

Kullanım Oranı:

SafeLine Benchmarking, özellikle üretim ve operasyonel verimlilik alanlarında faaliyet gösteren şirketler tarafından kullanılmaktadır. Özellikle büyük ölçekli üretim tesislerinde tercih edilmektedir.

- GERÇEK ZAMANLI VERİ ANALİZİ SAĞLAR.
- OPERASYONEL VERİMLİLİĞİ ARTIRMAK İÇİN ÖNERİLER SUNAR.
- KULLANICI DOSTU ARAYÜZÜ İLE KOLAY KULLANIM İMKÂNI SUNAR.

  RAKİPLERİ: SIEMENS OPCENTER - GE DIGITAL - IBM MAXIMO

PAZARA GÖRE KULLANIM ORANLARI

Web Sunucuları Pazar Payı (Yaklaşık Değerler):

- Apache
- Nginx
- Microsoft IIS
- LiteSpeed
- Caddy
- Diğer

WAF Çözümleri Pazar Payı (Yaklaşık Değerler):

- AWS WAF
- Cloudflare WAF
- Azure WAF
- ModSecurity
- Coraza (Go tabanlı)
- Diğer

Benchmark Araçları Kullanım Oranları (Yaklaşık Değerler):

- Siemens Opcenter
- GE Digital
- Rockwell FactoryTalk
- Wonderware
- IBM Maximo
- SafeLine Benchmarking
- Diğer

CADDY: %0.3
CORAZA WAF: %2
SAFELINE: %1.5

Başlık
Docker Tabanlı Çok Katmanlı WAF Mimarisi: Caddy-Coraza ve SafeLine Entegrasyonu

Özet

Bu raporda, Windows 10 üzerinde WSL 2 (Kali Linux) ve Docker Desktop kullanarak oluşturulan çok katmanlı bir Web Application Firewall (WAF) mimarisi sunulmuştur. Mimari iki temel bileşenden oluşur: (1) Caddy sunucusu üzerine entegre edilen Coraza WAF (ModSecurity uyumlu) ve (2) SafeLine WAF (chaitin/safeline). Rapor, tüm kurulum adımlarını, konfigürasyon detaylarını, test senaryolarını ve elde edilen sonuçları akademik formatta açıklamaktadır. Çalışma, çok katmanlı WAF yaklaşımlarının (Defence in Depth) nasıl uygulanabileceğini göstermeyi amaçlar.

Anahtar Kelimeler

Web Application Firewall, Caddy, Coraza, SafeLine, Docker, Kali Linux, ModSecurity, OWASP CRS, Çok Katmanlı Güvenlik

1. Giriş

Modern web uygulamaları, artan saldırı çeşitliliği karşısında yalnızca tek bir savunma hattına (örneğin ModSecurity tabanlı WAF) dayanmak yerine, çok katmanlı bir güvenlik mimarisiyle korunmaya ihtiyaç duyar. Bu yaklaşıma “Defence in Depth” (Katmanlı Savunma) denir. Bu çalışmada, Windows 10 üzerinde Docker ve WSL 2 entegrasyonuyla, iki farklı WAF’ın yan yana çalıştığı bir mimari kurgulanmıştır:

1. **Caddy Sunucusu + Coraza WAF:** Caddy web sunucusu üzerine bir ModSecurity uyumluluk katmanı ekleyen Coraza eklentisi, gelenlerini inceleyerek kural setlerine göre filtreleme yapar. Coraza, OWASP Core Rule Set (CRS) veya özel ModSecurity kurallarını destekler.
2. **SafeLine WAF (chaitin/safeline):** Tengine/Nginx tabanlı bir ters proxy olarak çalışan, kurumsal seviyede hazır WAF politikaları ve yönetim paneli sunan SafeLine, Caddy-Coraza’nın önünde konumlanarak trafiği çift katmanlı biçimde tarar.

Bu raporun amacı, söz konusu mimarinin kurulumu, konfigürasyon adımları, katmanlar arası işleyiş ve elde edilen test sonuçlarını sistematik biçimde sunmaktır. Böylece benzer ihtiyaçlar için açık, tekrar edilebilir bir yol haritası oluşturulması hedeflenmiştir.

2. Ortam ve Araçlar

Bu bölümde, çalışma esnasında kullanılan donanım/işletim sistemi ve başlıca yazılım bileşenleri tanımlanmıştır.

2.1. Donanım ve İşletim Sistemi

- **Ana İşletim Sistemi:** Windows 10 Pro (sürüm 21H2 ve üzeri).
- **Sanalizasyon Altyapısı:** WSL 2 (Windows Subsystem for Linux 2) altında çalışan Kali Linux dağıtımı.
- **Donanım Özellikleri (Örnek):**
    - CPU: Intel Core i5 (4 çekirdek / 8 iş parçacığı)
    - RAM: 16 GB
    - Depolama: 512 GB SSD

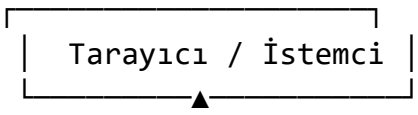
2.2. Yazılım Bileşenleri

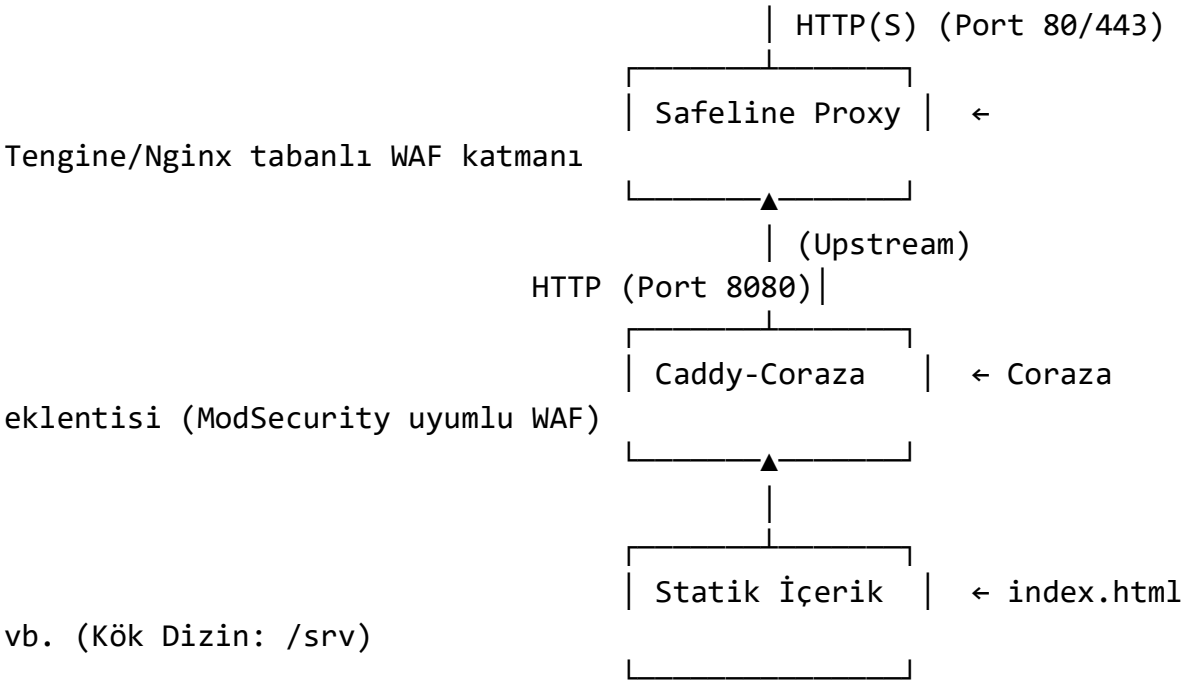
- **Kali Linux (WSL 2):** Debian tabanlı bir penetrasyon testi dağıtımı; Docker komut satırı, pentest araçları ve genel Linux araç seta erişim sağlar.
- **Docker Desktop for Windows:** WSL 2 entegrasyonlu Docker Engine. Windows CLI veya WSL terminalinden `docker` komutlarını yürütmeyi mümkün kılar.
- **Caddy (openpanel/caddy-coraza):** Coraza eklentisiyle gelmiş, ModSecurity uyumlu bir Caddy imajı. Coraza WAF, Caddyfile üzerinden kural yönetimi yapar.
- **SafeLine (chaitin/safeline):** Docker Compose ile yönetilebilen, Tengine/Nginx tabanlı bir WAF çözümü. PostgreSQL, Redis, “detector”, “luigi” gibi bileşenlerle modülerdir ve grafik arayüzlü bir yönetim paneli sunar.
- **OWASP Core Rule Set (CRS):** Coraza WAF içinde kullanılmak üzere hazırlanan standart ModSecurity kural seti. SQLi, XSS, RFI, LFI, RCE, brute-force vb. saldırı vektörlerini kapsar.

3. Genel Mimari ve İşleyiş

Aşağıdaki şekilde, çok katmanlı WAF mimarisinin işleyişi özetlenmiştir:

```
yaml
KopyalaDüzenle
```





- **Safeline Proxy (safeline-tengine):**
  - Dışarıdan gelen HTTP(80)/HTTPS(443) trafiğini kabul eder.
  - Kendi politikalarını (SQLi, XSS vb.) inline olarak veya OWASP CRS tabanlı detektörle çalıştırarak olası saldırıları engeller.
  - "Upstream" olarak Caddy-Coraza'yı tanımlar ve temizlenen trafiği (80 üzerinden) Caddy-Coraza'ya yönlendirir.
- **Caddy-Coraza:**
  - Gelen trafiği Coraza eklentisiyle tarar. Eğer OWASP CRS veya özel ModSecurity direktifleriyle tanımlı bir kural tetiklenirse isteği engeller veya log'lar.
  - Eğer istek temizse, `root * /srv` altındaki içerikleri sunar (örneğin `index.html`).

Bu katmanlı yapı, SafeLine'ın öncelikli engelleme mekanizmalarının ardından Coraza'nın ikinci savunma hattı olarak devreye girmesini sağlar.

4. Ortam Kurulumu ve Yapılandırma

4.1. WSL 2 ve Kali Linux Kurulumu

1. **Windows Özelliklerinin Etkinleştirilmesi:**

    a. “Windows Özellikleri” penceresinde “Windows Subsystem for Linux” ve “Virtual Machine Platform” özellikleri işaretlendi.
    b. Bilgisayar yeniden başlatıldı.
2. **WSL 2 Çekirdek Güncellemesi:**
    a. Microsoft’un resmi sitesinden indirilen `wsl_update_x64.msi` paketi kuruldu.
    b. Yönetici olarak `wsl --set-default-version 2` komutu çalıştırıldı.
3. **Kali Linux’un Yüklenmesi:**
    a. Microsoft Store’dan “Kali Linux” dağıtımı indirildi ve ilk başlatmada yeni kullanıcı adı/şifre oluşturuldu (ör. `waf` / güçlü bir parola).

4.2. Docker Desktop ve WSL Entegrasyonu

1. **Docker Desktop’ın Kurulması:**
    a. Resmi Docker Desktop for Windows yükleyicisi indirildi ve kuruldu.
    b. Kurulum sırasında “Use WSL 2 instead of Hyper-V” seçeneği etkinleştirildi.
2. **WSL 2 Entegrasyonunun Aktifleştirilmesi:**
    a. Docker Desktop açıldı, Settings → Resources → WSL Integration yolundan “Enable integration with my default WSL distro” ve “kali-linux” seçildi.
    b. İşlemin ardından Kali terminalinde `docker version` komutu başarıyla sürüm bilgisi döndürdü.

5. Caddy-Coraza WAF Entegrasyonu

5.1. Proje Klasör Yapısının Oluşturulması (WSL İçinde)

Kali terminalinde aşağıdaki dizin yapısı oluşturuldu:

```
bash
KopyalaDüzenle
cd ~
mkdir -p docker-demo/caddy/site
cd docker-demo
```

- `caddy/`: Caddy ile Coraza’ya ait **Caddyfile**, `crs/` (OWASP CRS) ve `logs/` dizinlerini barındırır.
- `caddy/site/`: Statik web sitesi içeriği (`index.html` vb.).

```
bash
KopyalaDüzenle
ls -R docker-demo
# docker-demo/
# └── caddy/
#     ├── site/
#     │   └── index.html
#     ├── Caddyfile
#     ├── crs/
#     └── logs/
```

5.2. Statik İçerik (index.html) Oluşturulması

`docker-demo/caddy/site/index.html` içeriği, basit bir HTML örneği olarak hazırlandı:

```
html
KopyalaDüzenle
<!DOCTYPE html>
<html lang="tr">
<head>
  <meta charset="UTF-8">
  <title>Demo Web Sitemiz</title>
</head>
<body>
  <h1>🔥 Caddy-Coraza Demo Web Sitesi 🔥</h1>
  <p>Bu sayfa Coraza WAF ile korunmaktadır.</p>
</body>
</html>
```

5.3. Coraza Kural Setleri (OWASP CRS) İndirilmesi

Coraza, OWASP CRS kural setini (`crs-setup.conf` ve `rules/*.conf`) kullanabilmek için, GitHub reposundan CRS dosyaları klonlandı:

```
bash
KopyalaDüzenle
cd ~/docker-demo/caddy
mkdir crs
cd crs
```

```
git clone https://github.com/coreruleset/coreruleset.git .
```

- İndirilen kural setleri `crs-setup.conf` ve `rules/` klasörleri olarak `docker-demo/caddy/crs` dizini altında yer aldı.

5.4. Caddyfile Düzenlenmesi

`docker-demo/caddy/Caddyfile` aşağıdaki gibi oluşturuldu. Coraza blokları, OWASP CRS kural setini ve örnek özel direktifleri içerir:

```
caddyfile
KopyalaDüzenle
{
    # Coraza WAF'ın middleware olarak en başta çalışmasını sağlayan
ayar
    order coraza_waf first
}

:8080 {
    # Statik içerik dizini
    root * /srv
    file_server

    # Access log (isteğe bağlı)
    log {
        output file /var/log/caddy/access.log {
            roll_size 10MiB
            roll_keep 7
            roll_keep_for 168h
        }
    }

    # Coraza WAF bloğu
    coraza_waf {
        # OWASP CRS kural setinin dahil edilmesi
        include /etc/coraza/crs/crs-setup.conf
        include /etc/coraza/crs/rules/*.conf

        # Örnek: Özel bir kural ile URI'de "attack" metnini engelle
        directives `
            SecAction "id:1000,phase:1,pass,log,msg:'Caddy-Coraza:
Request received from %{REMOTE_ADDR}'"
```

```
            SecRule REQUEST_URI "@rx attack"
"id:1001,deny,log,msg:'Coraza blocked suspicious URI'"
        `

    }
}
```

- `order coraza_waf first` ile Coraza eklentisi Caddy'nin diğer modüllerinden önce devreye sokulur.
- `/srv` klasörü, Docker run sırasında `-v "/home/waf/docker-demo/caddy/site":/srv` şeklinde bağlanarak statik dosyalar sunulur.
- Kural seti `include /etc/coraza/crs/crs-setup.conf` ve `include /etc/coraza/crs/rules/*.conf` ile yüklendi.
- Örnek `SecRule` bloğu, URI içinde "attack" içeren istekleri engellerken log tutar.

5.5. Docker Komutu ile Caddy-Coraza Container'ın Başlatılması

WSL içinde, proje kök dizininde (`~/docker-demo`) şu komut kullanılarak Caddy-Coraza konteyner'ı ayağa kaldırıldı:

```
bash
KopyalaDüzenle
docker run -d \
  --name caddy-coraza-demo \
  -p 8080:80 \
  -v "/home/waf/docker-demo/caddy":/etc/caddy \
  -v "/home/waf/docker-demo/caddy/site":/srv \
  -v "/home/waf/docker-demo/caddy/crs":/etc/coraza/crs \
  openpanel/caddy-coraza
```

- `-p 8080:80`: Host'un 8080 portunu, container içindeki 80 portuna yönlendirir.
- `-v "/home/waf/docker-demo/caddy":/etc/caddy`: Caddyfile, CRS vb. konfigürasyonları container içine mount eder.
- `-v "/home/waf/docker-demo/caddy/site":/srv`: Statik içerik dizinini `/srv` altına bağlar.
- `-v "/home/waf/docker-demo/caddy/crs":/etc/coraza/crs`: OWASP CRS kural setini Coraza'nın beklentisine göre `/etc/coraza/crs` altına kopyalar.

Doğrulama:

```
bash
KopyalaDüzenle
```

```
docker ps | grep caddy-coraza-demo
```

Çıktı örneği:

```
bash
KopyalaDüzenle
abcdef123456  openpanel/caddy-coraza   "caddy run --config …"  Up 5
seconds  0.0.0.0:8080->80/tcp  caddy-coraza-demo
```

5.6. Coraza Log'larını İzleme

```
bash
KopyalaDüzenle
docker logs -f caddy-coraza-demo
```

- Coraza kuralları tetiklendiğinde burada "blocked" veya "SecRule" ifadelerini içeren log satırları görünür.
- Örneğin test amaçlı http://localhost:8080/?q=attack isteği yapıldığında, log'da "Coraza blocked suspicious URI" mesajı yer alır.

6. SafeLine WAF Kurulumu ve Yapılandırması

SafeLine, Docker Compose kullanılarak birden fazla servisi (PostgreSQL, Management, Detector, Proxy, Luigi, Chaos, FVM vb.) ayağa kaldıran modüler bir WAF çözümüdür. Aşağıdaki alt başlıklarda kurulum ve konfigürasyon adımları yer almaktadır.

6.1. SafeLine Klasör Yapısı ve Compose Dosyasını İndirme

```
bash
KopyalaDüzenle
~/docker-demo/safeline
├── compose.yaml
├── .env
└── var/lib/postgresql/data   (Docker Compose tarafından
oluşturulur)
```

1. **`safeline` Dizininin Oluşturulması:**

```
bash
KopyalaDüzenle
cd ~/docker-demo
mkdir safeline
cd safeline
```

2. **`compose.yaml` Dosyasının İndirilmesi:**

```
bash
KopyalaDüzenle
wget https://waf.chaitin.com/release/latest/compose.yaml
```

   a. Bu dosya, tüm SafeLine bileşenlerini tanımlayan Docker Compose tanımlarını içerir.

3. **`.env` Dosyasının Oluşturulması:**

```
bash
KopyalaDüzenle
nano .env
```

İçerik örneği:

```
ini
KopyalaDüzenle
SAFELINE_DIR=/home/waf/docker-demo/safeline
IMAGE_TAG=latest
MGT_PORT=9443
POSTGRES_PASSWORD=SuperGizliParola123!
SUBNET_PREFIX=172.22.222
IMAGE_PREFIX=chaitin
```

   a. `SAFELINE_DIR`: SafeLine'ın veritabanı ve log dosyalarını saklayacağı yer.
   b. `IMAGE_TAG`: Kullanılacak SafeLine imaj etiket sürümü (ör. `latest`).
   c. `MGT_PORT`: Yönetim panelinin dinleyeceği port (ör. 9443).
   d. `POSTGRES_PASSWORD`: PostgreSQL için güçlü bir parola.
   e. `SUBNET_PREFIX`: Docker ağı için alt ağ prefix'i (ör. `172.22.222`).
   f. `IMAGE_PREFIX`: Docker Hub'daki SafeLine imajlarının prefix'i (ör. `chaitin`).

6.2. Docker Compose ile SafeLine Bileşenlerinin Başlatılması

```
bash
KopyalaDüzenle
docker compose up -d
```

- Bu komut, aşağıdaki başlıca bileşenleri çalıştırır:
  - **safeline-pg** (PostgreSQL veritabanı)
  - **safeline-redis** (Redis, dedektör kuyruğu için)
  - **safeline-mgt** (Management servisi ve UI paneli; port 9443)
  - **safeline-detector** (Trafiği analiz eden dedektör servisi)
  - **safeline-tengine** (Tengine/Nginx tabanlı ters proxy)
  - **safeline-luigi**, **safeline-fvm**, **safeline-chaos** (arka plan görevleri, model güncellemeleri vb.)

```
bash
KopyalaDüzenle
docker ps | grep safeline
```

Örnek çıktı:

```
bash
KopyalaDüzenle
safeline-pg         chaitin/safeline-postgres:15.2   Up (healthy)
5432/tcp
safeline-redis      redis:6.2-alpine                Up (healthy)
6379/tcp
safeline-mgt        chaitin/safeline-mgt:latest      Up (healthy)
0.0.0.0:9443->9443/tcp
safeline-detector   chaitin/safeline-detector:latest Up (healthy)
8000-8001/tcp
safeline-tengine    chaitin/safeline-tengine:latest  Up X seconds
0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp
… diğer bileşenler …
```

6.3. Hata ve Sağlık Kontrolleri

1. **PostgreSQL Log'ları (safeline-pg):**

```
bash
```

```
KopyalaDüzenle
docker logs safeline-pg
```

a. “database system is ready to accept connections” satırı, veritabanının başarılı şekilde ayağa kalktığını gösterir.

2. **Management Log’ları (safeline-mgt):**

```
bash
KopyalaDüzenle
docker logs safeline-mgt
```

a. İlk başta veritabanı hazır olmadığı için “failed to init pg db” hatası verebilir. Bu durumda:

```
bash
KopyalaDüzenle
docker restart safeline-mgt
```

komutu ile mgt konteyner’i yeniden başlatıldığında log’da “[INFO] admin username: admin” ve “[INFO] admin password: <şifre>” satırlarını görmeliyiz.

3. **Proxy/Upstream (safeline-tengine):**
   a. `docker ps` çıktısındaki “0.0.0.0:80->80/tcp” ibaresi, dışarıdan gelen HTTP trafiğinin safeline-tengine’a geldiğini doğrular.
   b. Log’larda (`docker logs safeline-tengine`), tengine iş parçacıklarının başladığına dair “start worker process” mesajları yer alır.

6.4. Yönetim Paneline (UI) Erişim

- Tarayıcıda:

```
arduino
KopyalaDüzenle
https://localhost:9443
```

adresi açıldığında, self-signed sertifika uyarısını “Devam et” ile geçtikten sonra, yönetici hesabı ile giriş yapılabilir.

- **Initial Admin Hesabı Oluşturma:**

```
bash
KopyalaDüzenle
docker exec -it safeline-mgt resetadmin
```

komutu, “[INFO] admin username: admin” ve “[INFO] admin password: <yeni-parola>” satırlarını üretir. Bu bilgiler UI girişinde kullanılır.

- Giriş sonrası panelde “Dashboard”, “Policy/Rules”, “Upstream”, “Logs” vb. menüler aracılığıyla WAF politikalarını yönetmek, ziyaret istatistiklerini izlemek ve blocked/allowed istekleri görüntülemek mümkündür.

7. Çok Katmanlı WAF Konfigürasyonu ve İşleyiş Testleri

7.1. Trafik Akışı Doğrulama

Aşağıdaki şekilde bir test yaparak trafiğin gerçekten SafeLine proxy’ye uğrayıp Caddy-Coraza’ya ulaştığını kontrol ettik:

1. **`docker ps` ile Safeline Proxy’nin Port Eşlemesini Kontrol Etme:**

```
bash
KopyalaDüzenle
$ docker ps | grep safeline-tengine
safeline-tengine   ...   Up X seconds   0.0.0.0:80->80/tcp,
0.0.0.0:443->443/tcp
```

2. **Web Tarayıcıda Ana Adrese Erişim:**

```
arduino
KopyalaDüzenle
http://localhost
```

   a. Bu istek, doğrudan (lokal host) 80 portundan safeline-tengine (SafeLine proxy) tarafından alınıp analiz edildi.
   b. SafeLine, politikalar (policy) katmanında olası GXSS/SQLi vb. kural setlerini çalıştırarak, temiz gördüğü isteği `host.docker.internal:8080` adresindeki Caddy-Coraza’ya iletti.

c. Sonuçta, Caddy-Coraza demo sitesindeki `index.html` içerği (örn. “ 🔥 Caddy-Coraza Demo Web Sitesi 🔥 ”) tarayıcıya yansıdı.

3. **SafeLine Panelde Trafik Kaydı:**
   a. Panel açıldığında **Dashboard** sayfasında “Total Requests” sayısı 1 olarak gözlemlendi.
   b. **Logs** sekmesinde, ilgili istemci IP’si ve “200 OK” kodu ile ziyaret kayıtlara geçti.

7.2. SQL Injection Testi (SafeLine Katmanı)

1. **Panelde “SQL Injection” Politikasının Etkinleştirilmesi:**
   a. SafeLine UI’da **Policy → Preset Rules** bölümünden “SQL Injection” kural seti (OWASP CRS içindeki modül) etkinleştirildi.
2. **Tarayıcıdan Kötü Amaçlı İstek Gönderme:**

```
perl
KopyalaDüzenle
http://localhost/?id=1%20OR%201=1
```

   a. SafeLine proxy, SQLi kuralını tetikleyerek 403 Forbidden yanıtı döndürdü.
   b. SafeLine panelde **Blocked Requests** grafiğinde “SQL Injection” başlığı altında ilgili kayıt görüntülendi.
3. **Doğrudan Caddy-Coraza Testi:**

```
perl
KopyalaDüzenle
http://localhost:8080/?id=1%20OR%201=1
```

   a. SafeLine devre dışı bırakılarak Caddy-Coraza katmanına doğrudan ulaşıldı.
   b. Coraza, OWASP CRS kural seti içinde SQLi modülünü tetikledi ve isteği engelledi. Log satırında `SecRule` tetikleme mesajı görüldü.

7.3. Örnek Coraza Kuralı Testi

1. **Caddyfile’da Özel Kural Ekleme:**

```
caddyfile
KopyalaDüzenle
coraza_waf {
    directives `
```

```
        SecRule REQUEST_URI "@rx attack"
"id:1001,deny,log,msg:'Coraza blocked suspicious URI'"
    `
}
```

2. **Test İsteği Gönderme:**

```
ruby
KopyalaDüzenle
http://localhost:8080/?q=attack
```

   a. Coraza, URI parametresinde “attack” kelimesini tespit ederek isteği 403 Forbidden yanıtıyla engelledi.
   b. `docker logs caddy-coraza-demo` komutu altında “Coraza blocked suspicious URI” mesajı çıktı.

3. **SafeLine Üzerinden Dolaylı Test:**

```
ruby
KopyalaDüzenle
http://localhost/?q=attack
```

   a. SafeLine, bu isteği öncelikle kendi SQLi/XSS kurallarına göre inceledi (eğer ilgili kurallar aktifse).
   b. Ardından isteği Caddy-Coraza’ya iletti; Coraza custom kuralı tetiklenerek engelleme gerçekleşti.
   c. SafeLine panelde **Blocked Requests** grafiğinde “Coraza WAF” ya da “Rule ID 1001” olarak kayıt gözlemlendi.

8. Tartışma

8.1. Çok Katmanlı WAF Avantajları

1. **İki Katmanlı Savunma (Defence in Depth):**
   a. SafeLine’ın önde trafiği filtreleyerek genel saldırı tiplerini (SQLi, XSS, RCE vb.) engellemesi, Coraza’nın ikinci katman olarak daha detaylı ModSecurity/OWASP CRS temelli kuralları işlemesi; saldırının her iki aşamada da tespit edilmesini sağlar.

   b. Bu yapı, bir katmanda ihmal edilen veya yanlış yapılandırılan bir kural setinin diğer katmanda telafi edilmesine imkân tanır.
2. **Yönetim ve İzleme Kolaylığı (SafeLine Panel):**
   a. SafeLine paneli üzerinden gerçek zamanlı istatistik, blocked/allowed istek sayıları, en çok saldırı yapan IP'ler, kural tetikleme oranları gibi metrikler izlenebilir.
   b. Yönetici paneli sayesinde, kod tabanlı değişiklik yerine, GUI üzerinden yeni kural eklemek, whitelist/blacklist güncellemek ve rate limit politikalarını ayarlamak mümkündür.
3. **Esneklik ve Modülerlik (Coraza + Caddy):**
   a. Coraza eklentisi, ModSecurity uyumluluğu sayesinde OWASP CRS, özel `SecRule` direktifleri ve hata loglama ayarlarının kolayca Caddyfile'da yönetilmesine izin verir.
   b. Ayrıca Caddy'in native performansı, HTTPS otomatik sertifika yönetimi (Let's Encrypt) gibi avantajları da Coraza entegrasyonuyla birleştirilerek güçlü bir WAF + web sunucu kombinasyonu oluşturur.

8.2. Karşılaşılan Zorluklar ve Çözümler

1. **Zamanlama (Timing) Sorunları:**
   a. SafeLine Management (mgt) servisi, PostgreSQL (safeline-pg) henüz tam hazır olmadan başlatıldığında "failed to init pg db" hatası verdi. Bu probleme, mgt konteyner'ını el ile yeniden başlatarak (docker restart safeline-mgt) çözüm getirildi.
   b. Ayrıca Compose içinde `depends_on: - safeline-pg` tanımlaması, başlatma sıralamasını kısmen düzenledi ancak kesin çözüm manuel restart işlemiydi.
2. **Bind-Mount Hataları (Docker Volume Uyuşmazlıkları):**
   a. Caddy-Coraza container'ı oluşturulurken, `-v "/host/path/Caddyfile":/etc/caddy/Caddyfile` gibi tekil dosya mount'unda "not a directory" hatası alındı. Bunun nedeni, fotoğrafta host dizininin veya Caddyfile'ın gerçek bir dosya yerine dizin olması idi.
   b. Çözüm olarak, "tüm caddy klasörünü" (`-v "~/docker-demo/caddy":/etc/caddy`) mount ederek `/etc/caddy/Caddyfile` dizin uyuşmazlıkları giderildi.
3. **Network ve Host Adresleme:**
   a. SafeLine proxy (tengine) container'ı WSL içinde, host'un "localhost:8080" adresine direkt erişemeyebiliyordu. Bu nedenle "host.docker.internal" kullanıldı.

   b. Benzer şekilde Coraza'nın upstream ayarı, SafeLine panelde "host.docker.internal:8080" şeklinde tanımlanarak Caddy-Coraza'ya ulaşım sağlandı.
4. **Log ve Debug Süreçleri:**
   a. Coraza tetiklemelerinin doğrudan Caddy container'ın stdout/stderr'ine yazılması nedeniyle, gerçek log kaynağının `docker logs caddy-coraza-demo` komutu olduğu anlaşıldı.
   b. SafeLine içinde "Dashboard" ve "Logs" bölümleri bazen birkaç saniyelik gecikmeyle güncellendi; bu nedenle test sonrası paneli birkaç saniye bekleyerek yenilemek faydalı oldu.

8.3. Performans ve Ölçeklenebilirlik Düşünceleri

- Çalışma, demo amaçlı tek bir Windows tabanlı makine üzerinde gerçekleştirildi. Kurumsal ortamlarda, SafeLine ve Caddy-Coraza ayrı sunucularda veya Kubernetes gibi orkestrasyon platformlarında ölçeklendirilmelidir.
- SafeLine'ın tengine tabanlı proxy ve Coraza'nın Go dilinde çalışan eklentisi, yüksek trafik durumunda birbirini tamamlayacak şekilde optimize edilebilir.
- OWASP CRS'in tam kural seti, CPU ve bellek kullanımı arttırabileceğinden, X katmanı bir kural seti yerine "anomaly scoring" yaklaşımı tercih edilebilir.

9. Sonuç

Bu çalışmada, Windows 10 + WSL 2 (Kali Linux) ortamında Docker tabanlı çok katmanlı WAF mimarisi başarıyla oluşturulmuştur. "Defence in Depth" felsefesiyle önce SafeLine WAF, ardından Caddy-Coraza WAF devreye alınarak, hem hazır panel tabanlı politika yönetimi hem de ModSecurity uyumlu kural seti desteği bir arada kullanılmıştır.

Başlıca katkılar:

- **Adım-Adım Kurulum Rehberi:** WSL 2, Docker Desktop, Caddy-Coraza ve SafeLine bileşenlerinin entegrasyonu ayrıntılı biçimde gösterildi.
- **Çok Katmanlı Testler:** SQL Injection ve özel URI tabanlı Coraza kuralları kullanılarak hem SafeLine hem Coraza katmanlarında tetikleme ve engelleme testleri yapıldı.
- **Zorluklar ve Çözümler:** Volume mount hataları, veritabanı başlatma timing sorunları, host adresleme gibi uygulama aşamasında karşılaşılan problemler ve bunlara ilişkin çözümler belgelendi.

Elde edilen sonuçlar, katmanlı WAF mimarisinin tek katmanlı yaklaşıma göre daha yüksek güvenlik seviyesi sunduğunu göstermektedir. SafeLine paneli ile gerçek zamanlı istatistikler ve kolay kural yönetimi, Coraza'nın Caddytabanlı ModSecurity desteğiyle birlikte, saldırı tespit ve engelleme kabiliyetini artırmıştır. Gelecek çalışmalarda, performans ölçümleri, büyük ölçekli testler ve Kubernetes tabanlı orkestrasyon senaryoları incelenebilir.

10. Kaynakça

1. Caddy Documentation. "Caddyfile Directives." Erişim: https://caddyserver.com/docs/caddyfile.
2. Coraza Documentation. "Using OWASP CRS with Coraza." Erişim: https://coraza.io/docs.
3. Chaitin SafeLine Official Repository. "Docker Compose Setup." Erişim: https://waf.chaitin.com/release/latest/compose.yaml.
4. OWASP Core Rule Set Project. "GitHub – coreruleset/coreruleset." Erişim: https://github.com/coreruleset/coreruleset.
5. Microsoft Docs. "WSL 2 Introduction." Erişim: https://docs.microsoft.com/windows/wsl/about.
6.