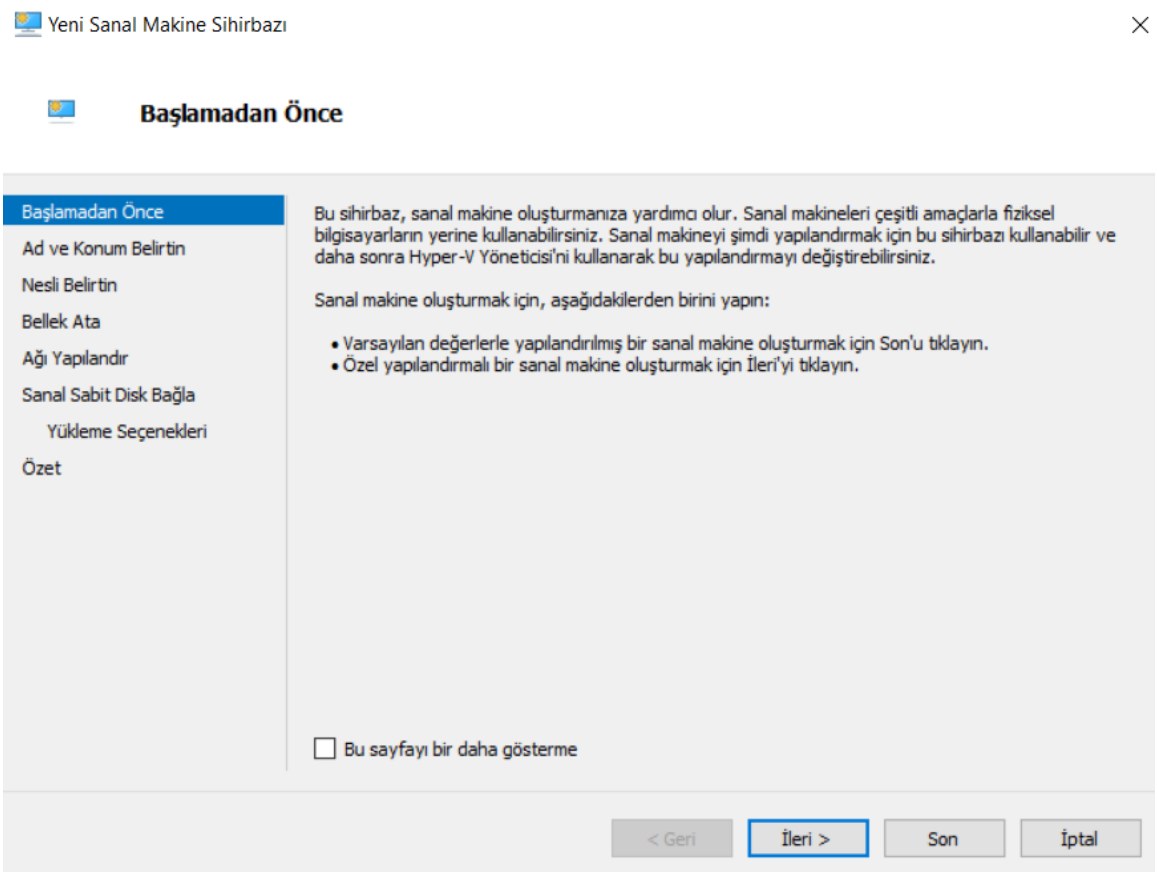
Freebsd işletim sistemine sıfırdan firewall kurulumu

Sanal Freebsd makinesi kurulumu

WMWare , VirtualBox veya Hyper-V gibi sanallaştırma yazılımları kullanabilirsiniz. Ben Hyper-V yi kullanıyorum.

Hyper-V nin kurulumunu yaptıktan sonra sol üst köşede "eylem < yeni < sanal makine" diyerek bu ekrana gelmeliyiz.



Burada "Ağı yapılandır " kısmına kadar kendi cihaz özellikleriniz ve istekleriniz doğrultusunda geliniz. Ağı yapılandır kısmında "bağlantı" kısmını **external** olarak ayarlamanızı ve yükleme seçenekleri kısmında Önyüklenebilir CD/DVD-ROM'dan işletim sistemi yükle diyerek , Görüntü dosyası (iso) seçeneğini seçip

iso dosyanızın konumunu buradan seçebilirsiniz.

SSH Bağlantısı

Öncelikli olarak kullanım kolaylığı sunması açısından ssh bağlantısını aktif ettim. Bunu sshd_config dosyasından yapacağız.

ee /etc/ssh/sshd_config
port 22
PermitRootLogin YES

Bu dosyanın sahipliği ve izinleri ise şu şekildedir :

```
root@samo:~ # ls -l /etc/ssh/sshd_config
-rw-r--r--  1 root wheel 3314 Sep 18 11:39 /etc/ssh/sshd_config
```

Artık daha kolay bir kullanım sağladık. Putty , WinSCP veya terminal gibi ekranlarda daha kolay bir şekilde işlerimizi yapacağız.

DHCP kurulumu ve aktif edilmesi

Kurulum aşamasında sizden EXTERNAL bağlantı türünü seçmenizi istemiştim , şimdi ise yeni bir ağ arayüzü oluşturup bunu da INTERNAL olarak seçmeliyiz. Bundan önce **dhcp (Dynamic Host Configuration Protocol)** yi açıklayalım. Dhcp , ağdaki cihazlara otomatik ip adresleri ve diğer yapılandırma bilgilerini atayan bir ağ protokolüdür. Bu ptorokol , ağ yöneticilerinin IP adreslerini manuel atama gerekliliğini ortadan kaldırarak ağ yönetimini basitleştirir , güvenli ip dağıtımını sağlar ve ağ segmentasyonu gibi konularda yardımcı olur.

ping 8.8.8.8 komutu ile öncelikle internet erişiminin olduğundan emin ol.
ifconfig komutu ile hn0 ağ arayüzünün ip adresinin olduğundan emin ol

Freebsd makinenizi kapatın ve Hyper-V ekranına gelin. Freebsd cihazınıza sağ tıklayarak ayarlar kısmına gelin
Donanım ekle < Ağ Bağdaştırıcısı diyin ve ekle butonuna tıklayın. Sanal anahtar kısmında **INTERNAL** seçin ve uygula , tamam diyin.

Artık LAN da bulunan cihazlarla iletişime geçebileceğimiz bir ağ arayüzümüz var.
Şimdi bu ağ arayüzüne ip adresi verelim , komut satırına gelin ve şu komutları çalıştırın. (ip adresini ve subnet mask ı kendinize göre ayarlayabilirsiniz)

ifconfig hn1 192.168.20.1 netmask 255.255.255.0

ifconfig hn1 up

Kalıcı olmasını sağlamak için ise

ee /etc/rc.conf a girerek şu komutu yazın :
ifconfig_hn1="inet 192.168.20.1 netmask 255.255.255.0"

hemen ardından komut ekranına " **service netif restart "** yazarak yaptığımız değişiklikleri etkinleştirin.

Olası hatalar:

Yeni bir ağ arayüzü eklenildiğinde diğer (hn0) ağ arayüzünün ip adresinin görülmemesi veya internet kesintisi gibi durumlar olabilir. Bu durumda öncelikle

ping 8.8.8.8 yaparak internetinizi test edin
ifconfig komutu ile tüm ağ arayüzlerinde ip adresinin olup olmadığını kontrol edin

netstat -rn komutu ile varsayılan ağ geçidinin doğruluğunu kontrol edin

```
root@samo:~ # netstat -rn
Routing tables

Internet:
Destination        Gateway            Flags     Netif Expire
default            192.168.10.1       UGS         hn0
```

Varsayılan ağ geçidinden kaynaklı bir hata alıyorsanız

ee /etc/pf.conf dosyasına bunları yazınız

ifconfig_hn1="inet 192.168.10... netmask 255.255.255.0"

defaultrouter="192.168.10.1"

dhcp servisinin sorunsuz ve cihazla beraber otomatik açılmasını sağlamak için bu komutların **/etc/rc.conf**

da mevcut olduğundan emin olun.

dhcpd_enable="YES"

dhcpd_flags="-q"

dhcpd_conf="/usr/local/etc/dhcpd.conf"

dhch_ifaces="hn1"

dhcpd_withumask="022"

ifconfig_hn0="inet 192.168.10.158 netmask 255.255.255.0"

ifconfig_hn1="inet 192.168.50.1 netmask 255.255.255.0"

defaultrouter="192.168.10.1"

Şimdi dhcp sunucusunu indirelim

pkg install isc-dhcp44-server

ee /usr/local/etc/dhcpd.conf

subnet 192.168.20.0 netmask 255.255.255.0 {

option routers 192.168.20.1;

option subnet-mask 255.255.255.0;

option domain-search "mydomain.local";

option domain-name-servers 8.8.8.8, 8.8.4.4;

option time-offset -18000;

range 192.168.20.10 192.168.20.100;

}

default-lease-time 600;

max-lease-time 7200;

Bu komutları kayıt edip çıktıktan sonra komut satırına

service isc-dhcpd start	yazarak servisi başlatabilirsiniz
service isc-dhcpd status	yazarak ise çalışıp çalışmadığını kontrol edebilirsiniz

NAT

hn0 ağ arayüzü internete erişebiliyor ama şimdilik hn1 in internete erişimi yok. Bunu da ancak NAT(Network Address Translation) ile yapabiliriz. NAT , TCP/IP ağındaki bir bilgisayarın yönlendirme cihazı ile başka bir ağa çıkarken adres uzayındaki bir IP ile yeniden haritalandırma yaparak IP paket başlığındaki ağ adres bilgisini değiştirme sürecidir.

pfctl -e
ee /etc/pf.conf

```
ext_if = "hn0"    # Dış arayüz

int_if = "hn1"    # İç arayüz


# NAT ayarları

nat on $ext_if from $int_if:network to any -> ($ext_if)


# Temel kurallar

block in all

pass out all

pass in on $int_if from any to any

pass in on $ext_if proto tcp from any to any port 80
```

Paket filter

etc nin altında bulunan pf.conf dosyası Packet Filter (PF) firewall'unun yapılandırma dosyasıdır. PF, ağ trafiğini yönetmek ve güvenlik sağlamak için kullanılan güçlü bir paket filtreleme sistemidir.

/etc/pf.conf dosyası, PF firewall'unun davranışını tanımlamak için kullanılır. Bu dosyada ağ trafiği üzerinde hangi kuralların uygulanacağı ve hangi paketlerin geçmesine izin verileceği ya da engelleneceği belirlenir.Pf.conf da **NAT** kurallarını yazarken block in all dedik bu da pass yapmadığımız tüm portların kapalı olacağı anlamına gelir. Bu yüzden gerek E2guardian gerek diğer programların kullanacağı portlar için pf.conf a özel kurallar girip izin vermek gerekir . Okuma işlemi yukarıdan aşağıya doğru gerçekleştiği için pass kurallarını block in all un altına koymamız daha sağlıklı olur.
/etc/rc.conf a packet filter ı tanımlamak lazım

pf_enable="YES"

pf_rules="/etc/pf.conf"

pflog_enable="YES"

E2guardian

Dansguardian'ın birçok iyileştirme ve hata düzeltmesi içeren bir dalı olan e2Guardian , Squid veya Oops gibi başka bir önbelleğe alma proxy'siyle birlikte çalışan bir web içeriği filtreleme proxy'sidir.

Belirli web sitelerine veya içeriğe erişimi engellemek veya kısıtlamak için kullanılabilir. Bu, yetişkin içerikleri, şiddet içeren veya zararlı içerikleri filtrelemek için sıkça kullanılır.

Kullanıcıların belirli kategorilere göre web sitelerini erişimlerine göre ayırmak için kategorik filtreleme uygulayabilir.
Farklı kullanıcı grupları için farklı filtreleme politikaları tanımlamak mümkündür. Bu, okul ortamında öğretmenlerin ve öğrencilerin farklı erişim izinlerine sahip olmasını sağlar.

Kullanıcıların veya grupların internet kullanımını izleyerek, hangi sitelerin ziyaret edildiğini kaydedebilir.

Kullanıcı etkinlikleri hakkında ayrıntılı raporlar oluşturabilir ve log dosyaları tutabilir. Bu, yönetim için önemli bir araçtır.

Hangi kullanıcıların hangi web sitelerini ziyaret ettiğini görmek için kullanılabilir.

e2guardian ın kurulumunu github dan çekerek yapacağız , daha güncel bir şekilde kullanmak için bunu yapıyoruz

pkg update
pkg upgrade
cd /root
pkg install git cmake gmake gcc pkgconf openssl pcre libevent automake autoconf libtool
git clone https://github.com/e2guardian/e2guardian.git
echo 'squid_enable=yes' | tee -a /etc/rc.conf
cd e2guardian

./autogen.sh
./configure --with-logdir=/var/log --with-piddir=/var/run --disable-avastd --enable-clamd --disable-icap --disable-kavd --prefix=/usr/local --localstatedir=/var --mandir=/usr/local/man --disable-silent-rules --infodir=/usr/local/share/info/ --build=amd64-portbld-freebsd14.0 --host=amd64-portbld-freebsd14.0 --sysconfdir=/usr/local/etc/e2guardian CXX=c++ CXXFLAGS="-O2 -pipe -fstack-protector-strong -fno-strict-aliasing -std=c++11" LDFLAGS="-fstack-protector-strong -L/usr/local/lib" LIBS="-lssl -lcrypto" CPPFLAGS="-I/usr/local/include" CC=cc CFLAGS="-O2 -pipe -fstack-protector-strong -fno-strict-aliasing" CPP=cpp PKG_CONFIG=pkgconf PKG_CONFIG_LIBDIR="/root/pfSense/FreeBSD-ports/www/e2guardian/work/.pkgconfig:/usr/local/libdata/pkgconfig:/usr/local/share/pkgconfig:/usr/libdata/pkgconfig" build_alias=amd64-portbld-freebsd14.0 host_alias=amd64-portbld-freebsd14.0

gmake
gmake install
/usr/local/sbin/e2guardian -v
/usr/local/sbin/e2guardian -d -c /usr/local/etc/e2guardian/e2guardian.conf
/usr/local/sbin/e2guardian -Q -c /usr/local/etc/e2guardian/e2guardian.conf

#eğer herhangi bir hata aldıysanız ***tail -f /var/log/messages*** yazarak hata mesajını kontrol edebilirsiniz**.** Eğer gmake yapmıyorsa şunu yap

pkg install pcre

export PKG_CONFIG_PATH=/usr/local/libdata/pkgconfig:/usr/local/share/pkgconfig:/usr/libdata/pkgconfig

#bundan sonra tekrar konfigürasyon komutlarını yaz)

Şimdi e2guardian.conf dosyasını hazır hale getirmemiz gerek:

E2Guardian'ın SSL trafiğini şeffaf bir şekilde yönlendireceği port numarasını belirlememiz gerekiyor. Bu port üzerinden gelen HTTPS trafiği izlenebilir. Örneğin, ***transparenthttpsport = 8081*** ayarını yaparak bu portu kullanabiliriz.

Ayrıca, SSL trafiğinin şifrelenmesini etkinleştirmeliyiz. Bu ayar, HTTPS üzerinden gelen taleplerin güvenli bir şekilde işlenmesine olanak tanır. E2Guardian'ın çalışacağı kullanıcı ve grubu da belirlememiz ve bunu yapılandırmamız gerekir; genellikle "***daemon***" olarak tanımlanabiliriz.

Kullanacağı dil dosyalarının bulunduğu dizini belirtmeliyiz. IP adresinizi öğrenmek için ifconfig komutunu kullanabilir ve filtrelemek istediğiniz IP adreslerini buraya belirtebilirsiniz. Ayrıca, bu IP adreslerini hangi port üzerinden dinleyeceğinizi ***filterport*** komutu ile tanımlayabilirsiniz.

Sertifika oluşturmanız da beklenir; bu sertifikalar cacert.pem, cakey.pem, serverkey.pem ve generatedcerts gibi dosyalarla birlikte belirtilmelidir. Bu dosyaların amacı, E2Guardian'ın SSL trafiğini güvenli bir şekilde yönetmesi ve şifrelemesidir.

Kullanıcı kimlik doğrulamasının yapılabilmesi için authplugin olarak ip.conf dosyasını tanımlamamız gerekiyor; bu, ağ üzerindeki kullanıcıların güvenliğini artırır. E2Guardian ile IP adreslerini yasaklayabilir ve istisna IP adresleri tanımlayabilirsiniz. Bunun için ***bannediplist*** ve ***exceptioniplist*** dosyaları kullanılır.

Ayrıca, ters DNS arama işlemini de etkinleştirebilir veya kapalı tutabilirsiniz. Kapalı olduğunda istemci IP'lerinin DNS çözümlemesi yapılmaz, bu da performansı artırır. IP'lerde yaptığımız gibi, sitelerde ve URL'lerde de engelleme veya istisna yapabiliriz; bunu ***urllist*** ve ***sitelist*** tanımları ile gerçekleştirebiliriz.

E2Guardian'ın erişim loglarının saklanacağı ***access.log*** dosyası bulunmaktadır. Bu loglar, kullanıcı etkinliklerini takip etmek için kullanılır ve güvenlik olaylarının analizinde faydalıdır. Logların düzeyleri ve biçimleri vardır; loglevel tanımı ile logların ayrıntı seviyesini belirtebilir ve ***logfileformat*** ile log dosyalarının biçimlerini kaydedebiliriz. İstisna durumların loglanma durumu ise ***logexceptiohits*** komutuyla 0-4 arasında bir seviyede log tutulmasını sağlar.

E2Guardian'ın trafik istatistikleri vardır ve buradan ağ performansını analiz edebiliriz. İstatistiklerin toplanacağı zaman aralıklarını da dstatlocation ve dstatinterval komutları ile belirtebiliriz. ***contentscanner = '/path/clamdscan.conf'*** dosyası ile içerik taramak için ClamAV kullanılıyorsa, bu dosya burada tanımlanır ve zararlı içeriklerin tespitinde gereklidir.

ClamAV'ı etkinleştirebilir, E2Guardian ile iletişim kuracağı adresi tanımlayabilir ve dinleyeceği portu belirtebilirsiniz. access.log dosyasındaki gibi, ***clamav.log*** dosyasını da tanımlayabiliriz; bu, virüs tarama süreçlerinin takibi için önemlidir. Logların biçimini varsayılan olarak bırakabiliriz, ancak şu şekilde ayarlayabiliriz:

log.access.format = "{date} {client_ip} {url} {result} {size} {status} {user_agent} {group} {proxy_name} {cache_status} {error}"

Bu ayar, logların okunabilirliğini ve analiz sürecini artırır. İndirilen dosyaların yönetimi için de downloadmanager tanımlayabilir ve dosya indirme sürecini daha kontrollü hale getirebiliriz.

E2Guardian'ın aynı anda kaç tane bağlantıyı işleyebileceğini, tanımladığımız worker sayısı belirler. Tarayıcıdan gelen içeriklerin boyutunu da ***maxcontentfiltersize*** komutu ile ayarlayabiliriz; bu ayar, kaynak kullanımı ve performans üzerinde önemli bir etkiye sahiptir.

Ayrıca, sorun giderme süreçlerini kolaylaştırmak için ***logconnectionhandlingerrors = on*** komutunu da kullanabiliriz.

!!!

Daha önce de belirttiğimiz üzere kullanılan portu pf.conf da pass komutunu kullanarak izin vermemiz gerekir.

/usr/local/etc/rc.d/ dizinine gelip e2guardian.sh dosyasını oluşturmamız gerekir ve içerisine shell betiği yazarız. Sebebi ise e2guardian hizmetinin başlatılmasını ve durdurulmasını yönetmeyi sağlamak.
chmod +x /usr/local/etc/rc.d/e2guardian.sh komutuyla da çalıştırma izni veririrz

```
#!/bin/sh

# PROVIDE: e2guardian

# REQUIRE: LOGIN

# KEYWORD: shutdown

. /etc/rc.subr

name="e2guardian"

rcvar=e2guardian_enable

command="/usr/local/sbin/${name}"

pidfile="/var/run/${name}.pid"

required_files="/usr/local/etc/${name}/${name}.conf"

load_rc_config $name

        run_rc_command "$1"
```

E2guardian ın dosya yapısını incelediğimiz zaman filtreleme , içerik tarama , kimlik doğrulama gibi çeşitli işlemleri yönetmek için dosyalar mevcuttur. Bunlar şu şekildedir :

/usr/local/etc/e2guardian/

authplugin

Bu dizin E2guardian'ın kimlik doğrulama eklentilerini barındırır.E2guardian, kullanıcı kimliğini belirlemek ve ona uygun erişim hakları vermek için çeşitli yöntemler kullanır. Bu yöntemler ve konfigürasyonları bu dizindeki dosyalarda tanımlanır.

common.story

Bu dosya, ortak hikaye yapısını içerir. E2Guardian'da filtrelemeler yapılırken hikaye tabanlı filtreleme kullanılır. Bu dosya, farklı hikaye tabanlı filtreleme senaryolarını belirler. Bu senaryolar, kullanıcıların davranışlarını izlemek ve belirli kurallara göre filtrelemeler yapmak için kullanılır.

common.story.sample

Bu, common.story dosyasının örnek versiyonudur. Yapılandırmayı anlamak ve kendi yapılandırmanızı oluştururken örnek almak için kullanılır. Bu dosya, nasıl bir hikaye yapılandırması yapabileceğiniz konusunda yol gösterici olabilir.

contentscanners

Bu dizin, içerik tarayıcılarının konfigürasyon dosyalarını içerir. Örneğin, ClamAV antivirüs tarayıcıyı yapılandırmak için kullanılan dosya buradadır. İçerik tarayıcıları, zararlı yazılımları ve diğer tehlikeli içerikleri tespit etmek için kullanılır.

downloadmanagers

Bu dizin, dosya indirme yöneticilerini yapılandırmak için kullanılır. Dosya indirme işlemlerini kontrol altında tutmak ve belirli dosya türlerine kısıtlama getirmek için bu dosyalar üzerinden ayarlamalar yapılabilir.

e2guardian.conf

E2Guardian'ın ana yapılandırma dosyasıdır. Burada, E2Guardian'ın temel işlevleri, ağ arayüzleri, filtreleme politikaları, loglama ayarları ve SSL ayarları gibi birçok temel yapılandırma yapılır. Filtreleme yapılacak IP'ler, portlar ve protokoller bu dosyada tanımlanır. Sistemin nasıl çalışacağı ve hangi özelliklerin aktif olacağı bu dosya aracılığıyla belirlenir.

e2guardian.conf.sample

e2guardian.conf dosyasının örnek versiyonudur. Yapılandırma sürecinde rehber olarak kullanılır. Kendi e2guardian.conf dosyanızı oluştururken bu örnek dosyadaki ayarları baz alabilirsiniz.

e2guardianf1.conf / e2guardianf2.conf

E2Guardian'ın alternatif yapılandırma dosyalarıdır. Bazı özel kullanım senaryoları için farklı yapılandırma dosyaları gerekebilir. e2guardianf1.conf ve e2guardianf2.conf bu tür senaryolar için farklı ayarları içerir. Örneğin, belirli bir filtreleme türünü sadece bu dosyalardan biri için aktif hale getirmek istiyorsanız, bu dosyaları kullanabilirsiniz.

e2guardianf1.conf.sample / e2guardianf2.conf.sample

e2guardianf1.conf ve e2guardianf2.conf dosyalarının örnek sürümleridir. Alternatif yapılandırma için rehber olarak kullanılır.

preauth.story

E2Guardian'da kimlik doğrulama işlemi yapılmadan önce çalışan bir hikaye dosyasıdır. Preauth işlemleri, kullanıcının kimliğini doğrulamadan önce belirli filtreleri çalıştırmanızı sağlar. Bu dosya, kullanıcılar filtreleme işlemi yapılmadan önce belirli işlemlerin gerçekleştirilmesi gerektiğinde devreye girer.

preauth.story.sample

preauth.story dosyasının örnek sürümüdür. Bu dosya, kimlik doğrulama öncesi işlemleri yapılandırmak için örnek bir yapı sunar.

site.story

site.story dosyası, belirli sitelere yönelik hikaye tabanlı filtrelemeler için kullanılır. Kullanıcıların hangi sitelere erişip hangi sitelere erişemeyecekleri bu dosya üzerinden yapılandırılabilir.

site.story.sample

site.story dosyasının örnek versiyonudur. Filtreleme politikalarınızı oluştururken bu örnek dosya üzerinden kurgulayabilirsiniz.

/usr/local/etc/e2guardian/authplugins/

dnsauth.conf

DNS tabanlı kimlik doğrulama için kullanılan bir yapılandırma dosyasıdır. Kullanıcıların DNS üzerinden kimliklerinin doğrulanması ve ona göre filtre gruplarına atanması bu dosyada tanımlanır. Örneğin, belirli bir DNS sunucusundan gelen trafik farklı bir filtreye tabi tutulabilir.

dnsauth.conf.sample

dnsauth.conf dosyasının örnek versiyonudur. DNS tabanlı kimlik doğrulamayı nasıl yapılandıracağınızı anlamak için kullanılır.

ident.conf

Ident protokolünü kullanarak kimlik doğrulama işlemlerini yapılandırır. Kullanıcı kimliğini belirlemek ve buna göre filtreleme işlemleri yapmak için kullanılır. Bu dosya, kullanıcının IP adresine göre filtre gruplarına atanmasını sağlar.

ident.conf.sample

ident.conf dosyasının örnek sürümüdür. Ident protokolü ile kimlik doğrulama yaparken referans olarak kullanılabilir.

ip.conf

IP tabanlı kimlik doğrulama için kullanılan bir dosyadır. Bu dosyada, istemcinin IP adresine göre hangi filtre grubuna atanacağını belirleyebilirsiniz. IP adresleri ve filtre grupları arasında eşleştirme yapılır.

ip.conf.sample

ip.conf dosyasının örnek sürümüdür. IP tabanlı kimlik doğrulama yapılandırmasını bu dosyadan örnek alarak gerçekleştirebilirsiniz.

port.conf

Belirli bir port üzerinden kimlik doğrulama işlemlerini yapılandırır. Bu dosya, trafiğin geldiği port numarasına göre filtreleme işlemi yapılmasını sağlar. Örneğin, belirli portlardan gelen trafik özel bir gruba atanabilir.

port.conf.sample

port.conf dosyasının örnek sürümüdür. Port tabanlı kimlik doğrulama için örnek bir yapı sağlar.

proxy-basic.conf

Proxy sunucusu üzerinden temel kimlik doğrulama işlemleri için kullanılan dosyadır. Proxy sunucusu ile yapılan talepleri temel kimlik doğrulama mekanizmalarıyla kontrol eder ve filtre gruplarına atamalar yapar.

proxy-basic.conf.sample

proxy-basic.conf dosyasının örnek sürümüdür. Proxy üzerinden temel kimlik doğrulama yapılandırmasını bu dosya üzerinden örnek alabilirsiniz.

proxy-digest.conf

Proxy sunucusu üzerinden gelişmiş (digest) kimlik doğrulama işlemleri için kullanılan dosyadır. Digest kimlik doğrulama, temel kimlik doğrulamaya göre daha güvenli bir yöntemdir.

proxy-digest.conf.sample

proxy-digest.conf dosyasının örnek sürümüdür. Proxy sunucusu üzerinden gelişmiş kimlik doğrulama yapılandırmasını bu dosya üzerinden örnek alabilirsiniz.

proxy-header.conf

Proxy sunucusunun başlık bilgilerini kullanarak kimlik doğrulama işlemi yapılır. Proxy başlıkları üzerinde yapılan doğrulamalar ile kullanıcılara filtre grupları atanabilir.

proxy-header.conf.sample

proxy-header.conf dosyasının örnek sürümüdür. Proxy başlıklarına göre kimlik doğrulama işlemlerini yapılandırmak için referans alınabilir.

/usr/local/etc/e2guardian/contentscanners/

clamdscan.conf

ClamAV antivirüs tarayıcısını yapılandırmak için kullanılan dosyadır. E2Guardian, zararlı içerikleri tespit etmek için ClamAV ile entegre çalışır. Bu dosya, ClamAV tarayıcısının yapılandırmasını ve E2Guardian ile nasıl etkileşimde olacağını tanımlar.

clamdscan.conf.sample

clamdscan.conf dosyasının örnek versiyonudur. ClamAV tarayıcı ayarlarını yapılandırırken bu dosyadan referans alınabilir.

/usr/local/etc/e2guardian/downloadmanagers/

default.conf

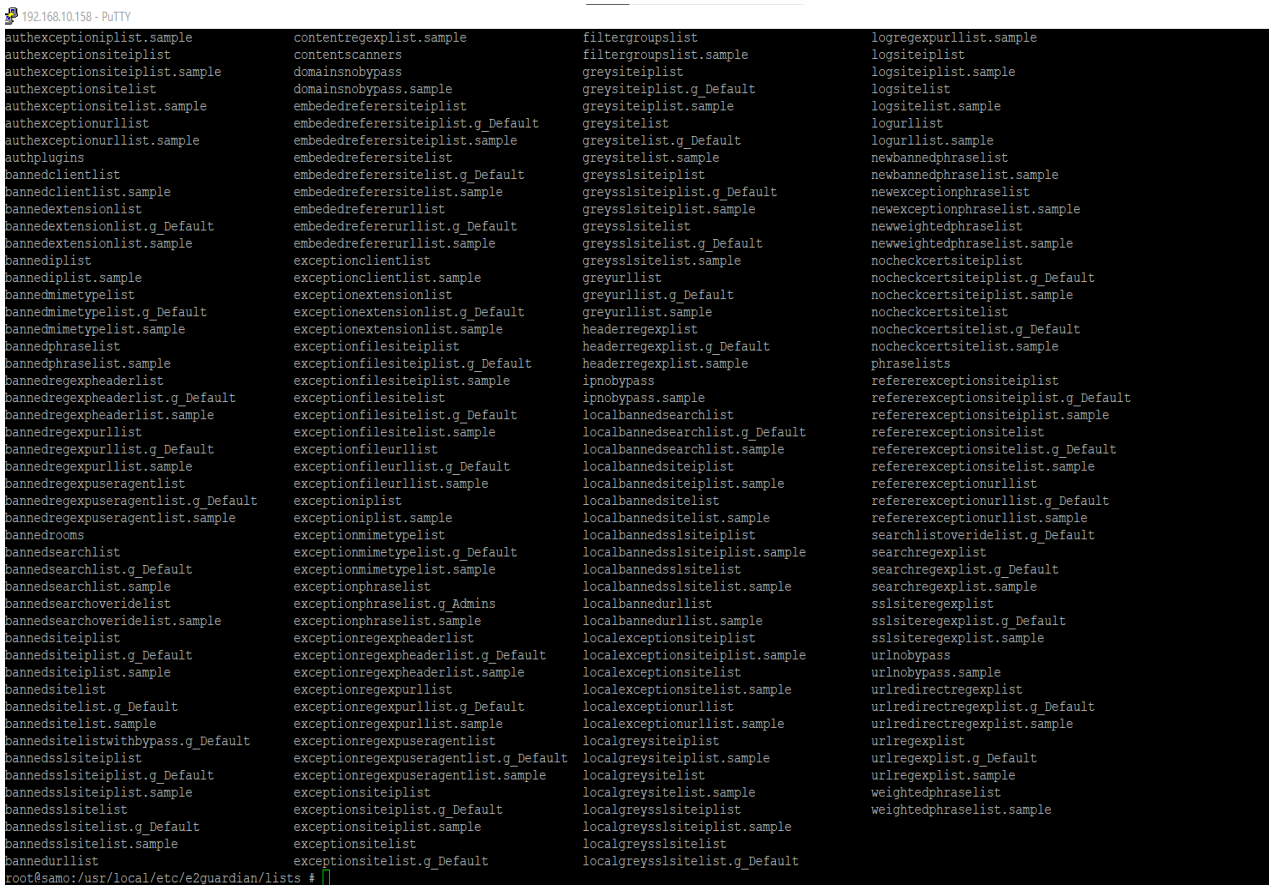
Dosya indirme işlemlerini kontrol eden varsayılan yapılandırma dosyasıdır. Bu dosya, hangi tür dosyaların indirilebileceği veya hangi dosya türlerine kısıtlama getirileceği gibi ayarları içerir.

default.conf.sample

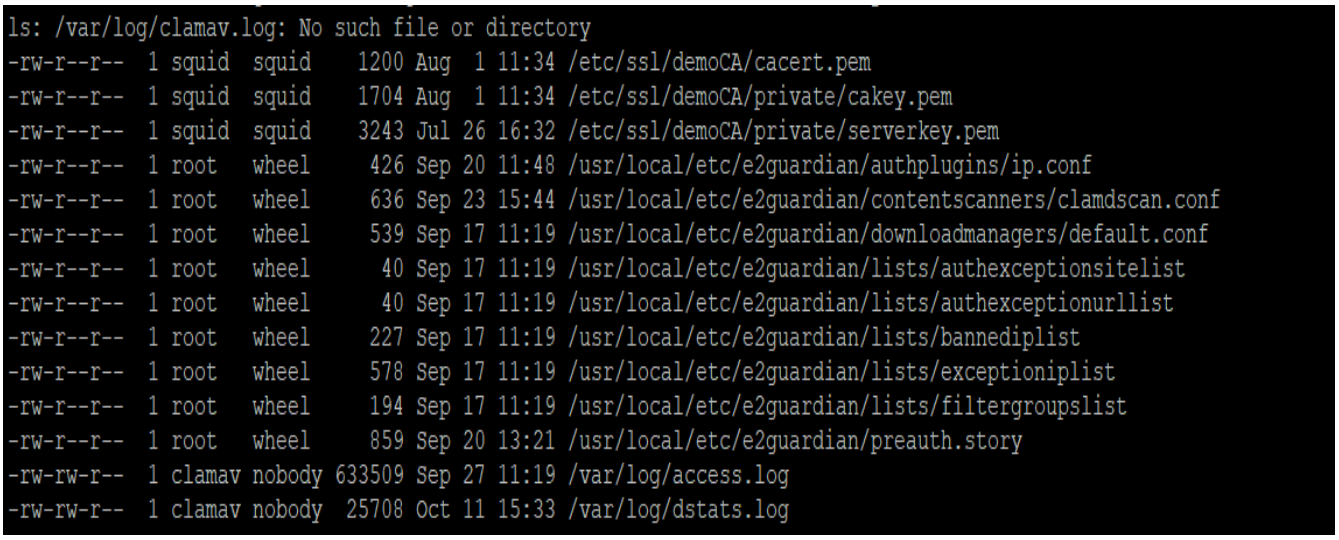
default.conf dosyasının örnek versiyonudur. Dosya indirme yöneticisinin nasıl yapılandırılacağına dair örnek sağlar.

/usr/local/etc/e2guardian/lists

Burada e2guardian ın kullanacağı listeler vardır ve şu şekilde kendi yazdığım dosyalar bu şekildedir:
Büyültmeniz halinde görseli daha net birşekilde görebilirsiniz.



E2guardian.conf un içine yazdığımız dosyaların sahiplikleri ve izinleri vardır
onları da şu şekilde dizayn etmek gerekir:



Bu ayarlamaları yaptıktan sonra access.log un çalışması hebüz beklenemez

tail -f /var/log/access.log yazmanız halinde herhangi bir çıktı vermeyecektir.

Log kayıtlarının tutulabilmesi için ağımıza bağlanan cihazda şu ayarlamaların yapılması gerekir:

Cihazında Proxy Ayarlarını Kontrol Edin

Windows cihazının proxy ayarlarının doğru yapılandırıldığından emin olun. e2guardian üzerinden filtreleme yapabilmek için cihazın e2guardian proxy'sini kullanması gerekir.

Internet Options (İnternet Seçenekleri) > Connections (Bağlantılar) > LAN settings (LAN ayarları) yolunu izleyin.

Proxy server (Proxy sunucusu) kutusunu işaretleyin ve e2guardian sunucunuzun IP adresi ile port numarasını girin (genellikle 8080 veya yapılandırmanıza bağlı olarak başka bir port).

Tekrardan logları incelediğimizde bir kısmını kırptığım görüntüye benzer bir çıktı vermesi gerekir (kişisel verilerin korunması amacıyla görselde çıktının tamamı mevcut değildir)

```
root@samo:/usr/local/etc/e2guardian/lists # tail -f /var/log/access.log
"2024.09.27 11:07:51","-","192.168.50.10","https://www.youtube.com:443","-",
"2024.09.27 11:08:51","-","192.168.50.10","https://www.youtube.com:443","-",
"2024.09.27 11:09:51","-","192.168.50.10","https://www.youtube.com:443","-",
"2024.09.27 11:10:51","-","192.168.50.10","https://www.youtube.com:443","-",
"2024.09.27 11:11:51","-","192.168.50.10","https://www.youtube.com:443","-",
"2024.09.27 11:12:51","-","192.168.50.10","https://www.youtube.com:443","-",
"2024.09.27 11:13:51","-","192.168.50.10","https://www.youtube.com:443","-",
"2024.09.27 11:14:01","-","192.168.50.10","https://www.youtube.com:443","-",
"2024.09.27 11:18:21","-","192.168.50.10","https://googleads.g.doubleclick.n
```

ICAP

ICAP (Internet Content Adaptation Protocol), istemci ve sunucular arasında web içeriği işlemesi yapabilen bir protokoldür. Temelde, web proxy sunucuları (örneğin, Squid) ile içerik filtreleme, antivirüs tarama, içerik modifikasyonu gibi hizmetleri sağlayan sunucular arasında veri alışverişini kolaylaştırmak için kullanılır. ICAP sayesinde, proxy sunucular gelen ve giden trafiği bir ICAP sunucusuna ileterek içerik üzerinde çeşitli işlemler yaptırabilir.

ICAP'ın temel amacı, web proxy sunucularını zorlamadan içerik adaptasyonu ve işlemesini merkezi bir ICAP sunucusunda gerçekleştirmektir. Bu, aşağıdaki işlemleri kapsayabilir:

Antivirüs taraması (zararlı içerik taraması)

Reklam kaldırma veya ekleme

İçerik filtreleme (zararlı veya uygunsuz içeriklerin engellenmesi)

Veri sıkıştırma veya modifikasyon

URL yeniden yazımı

C-ICAP Nedir?

C-ICAP, ICAP protokolünü uygulayan ve çeşitli hizmetleri destekleyen bir ICAP sunucusu yazılımıdır. Genellikle proxy sunucular (örneğin, Squid) ile entegre çalışarak içeriği işlemek ve belirli servisler sağlamak için kullanılır.

C-ICAP'ın başlıca işlevleri şunlardır:

Antivirüs taraması: C-ICAP, web trafiği üzerindeki dosyaları ve içerikleri antivirüs motorları (örneğin, ClamAV) ile tarayabilir. Proxy sunucusundan gelen içerik, c-icap sunucusuna yönlendirilir ve zararlı içerik

olup olmadığı denetlenir.

İçerik filtreleme: Zararlı veya uygunsuz içeriğin (örn. yetişkin içerikler, kötü amaçlı yazılımlar) kullanıcıya ulaşmadan önce engellenmesi sağlanabilir.

Modifikasyon işlemleri: İçerik modifikasyonu yapılabilir; örneğin, web sayfalarına banner eklemek, URL'leri yeniden yazmak veya reklamları kaldırmak gibi işlemler c-icap ile yapılabilir.

C-ICAP Kullanım Senaryoları

C-ICAP genellikle aşağıdaki durumlar için kullanılır:

Web Proxy Entegrasyonu: Squid veya başka bir proxy sunucusu ICAP ile entegre edilerek, isteklerin bir kısmı ICAP sunucusuna yönlendirilir ve burada filtreleme, tarama veya modifikasyon işlemleri gerçekleştirilir. Örneğin, Squid bir kullanıcının internet trafiğini alıp, ICAP sunucusuna gönderir, c-icap sunucusu da bu trafiği tarar ve güvenli hale getirir.

Antivirüs Taraması: C-ICAP, ClamAV gibi bir antivirüs motoruyla entegre edilerek, proxy sunucusu üzerinden gelen dosyaların ve web trafiğinin zararlı yazılım olup olmadığını kontrol eder. Zararlı içerik tespit edilirse kullanıcıya erişimi engelleyebilir.

Veri Filtreleme ve Uygulama Güvenliği: C-ICAP, web proxy sunucularıyla birlikte çalışan bir içerik filtreleme çözümü olarak kullanılabilir. URL'leri veya içerikleri engelleyebilir, güvenli olmayan web sitelerini kara listeye alabilir.

ICAP ve Squid Örneği

Squid gibi bir proxy sunucusu, gelen web trafiğini c-icap sunucusuna ICAP protokolü üzerinden yönlendirir. C-ICAP sunucusu, bu trafiği tarar, içerik modifikasyonu yapar veya isteğe göre engeller, ardından bu trafiği Squid sunucusuna geri iletir ve kullanıcıya yönlendirir. Örneğin:

Kullanıcı bir web sitesine erişim isteğinde bulunur.

Squid bu isteği alır ve c-icap sunucusuna gönderir.

C-ICAP, isteği antivirüs ile tarar, uygunsuz içeriği filtreler veya başka bir işlem yapar.

Sonuç Squid'e geri iletilir ve kullanıcıya güvenli içerik sunulur.

Kurulumu

pkg update
pkg install gcc gmake curl openssl libevent glib pkgconf
cd /usr/local/src

fetch https://github.com/c-icap/c-icap/archive/refs/tags/0.5.4.tar.gz

tar -xzf 0.5.4.tar.gz

cd c-icap-0.5.4
./configure --prefix=/usr/local/c-icap --enable-ssl
make
make install
cp /usr/local/c-icap/etc/c-icap.conf.sample /usr/local/c-icap/etc/c-icap.conf (dosyaların konumuna ve kurulumunuzun faklılığınıza göre path ler değişebilir)

ee /usr/local/c-icap/etc/c-icap.conf gerekli ayarlamaları yapınız
/usr/local/c-icap/sbin/c-icap -f /usr/local/c-icap/etc/c-icap.conf başlatmak için

#otomatik başlatma için bu shell komutunu verdiğim dizine yazınız

ee /usr/local/etc/rc.d/c-icap

```
#!/bin/sh

# PROVIDE: c-icap

# REQUIRE: LOGIN

# KEYWORD: shutdown

. /etc/rc.subr

name="c-icap"

rcvar=c_icap_enable
```

command="/usr/local/c-icap/sbin/c-icap"

pidfile="/var/run/${name}.pid"

required_files="/usr/local/c-icap/etc/c-icap.conf"

load_rc_config $name

run_rc_command "$1"

chmod +x /usr/local/etc/rc.d/c-icap çalıştırılabilir hale getirir

echo 'c_icap_enable="YES"' >> /etc/rc.conf

service c-icap start

curl http://localhost:1344/ C-ICAP'ın doğru çalıştığını test etmek için:

```
root@samo:/usr/local/etc/e2guardian/lists # service c-icap start
clamav_enable: YES -> YES
freeradius3_enable: YES -> YES
clamav_enable: YES -> YES
freeradius3_enable: YES -> YES
c_icap already running?  (pid=28556).
```

Konfigürasyon Dosyası

Icap ın konfigürasyob dosyasında ise kullanmak istediğiniz ip adresini (ben localhost da çalıştırıyorum) veriniz. Port olarak ise 1344 portunu tavsiye ediyorum.

Pid dosyasının konumunu veriniz , bu dosya sunucunun çalıştığı süreçle ilgili bilgileri içerir.

Yönetimsel komutların sunucuya iletilmesi için CommandSocket pathini veriniz

Yanıt alınamayan isteklerde sunucunun isteği iptal etmesi için süreyi giriniz (timeout)

Performansı artırmak ve bağlantı yönetimini optimize etmek için bir bağlantıda işlenebilecek maksimum keep-alive isteğinin sayısını ayarlayınız.

Süre dolduğunda bağlantının kapatılması gerekir. Keep-alive bağlantılarında sunucunun bekleyeceği max süreyi giriniz

Hızlı bir yanıt verme süresi sağlamak için sunucu başlatıldığında oluşturulacak başlangıç sunucu sayısını

giriniz. (MaxServers)

Ani yük artışlarında hızlı bir şekilde yanıt verilebilmesi için her zaman hazır bekletilmesi gereken minumum iş parçacığı sayısını giriniz (MinSpareThreads)

Sunucunun yük altında performansını arttırmak için her zaman hazır bekletilmesi gereken maksimum iş parçacığı sayısını giriniz (MaxSpareThreads)

İşlemci kaynaklarının verimli bir şekilde kullanılmasını sağlamak için her çocuk sunucunun işleyebileceği maksimum iş parçacığı sayısını giriniz (ThreadsPerChild)

Sunucu süreçlerinin yönetimini optimize etmek için çocuk sunucunun işleyebileceği maksimum istek sayısını giriniz (MaxRequestsPerChild)

Sorun durumunda iletişim kurmayı kolaylaştırmak amacıyla sunucu yöneticisinin iletişim bilgilerini içeren ayarı yapınız (ServerAdmin)

Sununun tanınabilirliği için tanımlayıcı adını giriniz (ServerName)

Geçici dosyaların yönetimini kolaylaştırmak için sunucu tarafından kullanılacak geçici dosyaların saklanacağı dizini giriniz (TempDir)

Bellek yönetimini optimize etmek için yönetilebilecek maksimum bellek nesne boyutunu giriniz (MaxMemObject)

Yüksek seviye , daha ayrıntılı bilgi sağlaması ve sorun gidermesi amacıyla sunucunun hata ayıklama seviyesini giriniz (DebugLevel)

Performans arttırmak amacıyla istemcilerin çoklu istekleri tek bir TCP bağlantısı üzerinden göndermesine izin verilebilir (Pipelining)

Hatalı istemcilerin desteklenip desteklenmeyeceni belirlemek için bu komutu kullanabiliriz (SupportBuggyClients)

Sunucunun genişletilebilirliğini sağlamak amacıyla C-ICAP modüllerinin bulunduğu dizin vardır(ModulesDir)

Şablonların etkin bir şekilde kullanmayı sağlamak amacıyla C-ICAP tarafından kullanılan şablon dosyaları TemplateDir in uzantısındadır.

Dosya türlerinin tanınmasını sağlamak amacıyla sunucu tarafından büyücü dosya kullanılabilir(LoadMagicFile)

İletişim güvenliğini sağlamak amacıyla uzaktan proxy kullanıcı başlığının kodlanması sağlanılabilir(RemoteProxyUserHeaderEncoded)

Sunucu üzerindeki tüm aktivitelerin izlenebilmesi için ServerLog dosyasına loglar kaydedilebilir

Erişim log dosyalarını ise Access.log a tanımlayabiliriz. (E2guardian'la çakışma yaşanmaması için /var/log/c-icap/access.log olarak tanımlayabiliriz)

FreshClam

Açık kaynaklı bir antivirüs yazılımı olan ClamAV'nin (Clam AntiVirus) bir bileşenidir. ClamAV, özellikle kötü amaçlı yazılım ve virüs tespiti için kullanılan, platformlar arası bir antivirüs motorudur. FreshClam ise bu motorun virüs imza veritabanını güncel tutmak için kullanılan bir araçtır.

Görevi ise ClamAV'nin etkili bir şekilde çalışabilmesi için virüs imzalarının (yani, bilinen virüs ve kötü amaçlı yazılımların özelliklerinin yer aldığı veritabanları) güncel olması gerekir. FreshClam, bu imza dosyalarını ClamAV'nin veritabanına indirir ve düzenli olarak günceller. Bu, yeni çıkan tehditlere karşı sistemin korunmasını sağlar.

Özellikleri :

Virüs İmza Güncellemesi: FreshClam, ClamAV'nin kötü amaçlı yazılımları tespit edebilmesi için virüs imza veritabanını sürekli günceller. ClamAV, bu veritabanı aracılığıyla kötü amaçlı yazılımları tespit eder ve kaldırır.

Otomatik Güncelleme: FreshClam, ClamAV veritabanını düzenli aralıklarla otomatik olarak güncelleyebilir. Bu, antivirüs motorunun her zaman en yeni tehditlere karşı koruma sağlamasını garanti eder. FreshClam yapılandırma dosyasında, güncellemelerin ne sıklıkla yapılacağını ayarlayabilirsiniz.

Dağıtılmış Aynalar: FreshClam, ClamAV imza dosyalarını ClamAV sunucularından ve yansıtılmış sunuculardan indirir. Bu, yükü azaltarak daha hızlı güncellemeler sağlar ve hizmetin sürekliliğini artırır.

Proxy ve Güvenlik Duvarı Desteği: FreshClam, proxy sunucular ve güvenlik duvarları arkasındaki sistemlerde çalışabilir. Bu, özellikle kurumsal ağlarda FreshClam'in güncellemeleri indirmesini kolaylaştırır.

Özel Veritabanları: FreshClam, yalnızca ClamAV'nin resmi virüs imza veritabanını değil, aynı zamanda özelleştirilmiş virüs imza veritabanlarını da destekler. Böylece yerel tehditlere veya organizasyonunuza özgü kötü amaçlı yazılımlara karşı koruma sağlayabilirsiniz.

pkg install clamav diyerek indirme işleminizi yapabilirsiniz.

Konfigürasyon dosyasında bir takım değişiklikler yapmanız gerekebilir; bu değişiklikler beklentilerinize ve amacınıza göre değişiklik gösterebilir.

Log dosyasının maksimum boyutunu belirleyebilirsiniz; bunu **megabayt** veya **kilobayt** cinsinden değerlerle ayarlayabilirsiniz.

Log kayıtlarına zaman damgası ekleyebilirsiniz ***LogTime*** komutu ile. Daha ayrıntılı loglama yapmasını isterseniz ***LogVerbose*** komutunu kullanarak daha ayrıntılı log kayıtları elde edebilirsiniz.

Sistem günlüğü kullanabilir ve günlük mesajlarının türünü de ayarlayabilirsiniz.

Pid dosyasının path'ini de girmeyi unutmayınız.

ClamAV ile entegre bir şekilde çalışmasını istediğimiz için sahipliğini ayarlarken clamav olarak ayarlamanızı tavsiye ederim. ***DatabaseOwner*** komutu ile de veritabanının sahipliğini de aynı şekilde clamav'a verebilirsiniz.

Virüs veritabanı sürümünü doğrulayabilir ve veritabanı güncellemeleri için kullanılacak ana alan adını da belirleyebilirsiniz.

Bağlantıların başarısız olması halinde kaç deneme yapılacağını veya veritabanı sunucusuna bağlanırken zaman aşımı değerlerinizi kendinize göre ayarlayabilir veya varsayılan ayarlara bırakabilirsiniz.

Veritabanı dosyaları için özel kaynaklar kullanabilirsiniz.

Checks komutu ile de günlük kontrol sayısını yazabilirsiniz; örneğin, 6 değerini girerseniz 4 saatte bir kontrol yapar. ***LocalIpAddress*** direktifi ile ClamAV'ın veritabanı güncelleyici bileşeninin veritabanlarını indirmek için kullanacağı istemci IP adresini girebilirsiniz.

NotifyClamd komutu sayesinde freshclam uygulamasının veritabanı güncellendiğinde ClamAV Daemon'a "*reload*" komutunun gönderilmesini sağlayabiliriz. Bu komut güncellenen veritabanının yüklenmesini ve kullanılmasını sağlar.

Debug komutu sayesinde libclamav kütüphanesinde hata ayıklama mesajlarını etkinleştirebilirsiniz. Bu direktif geliştiriciler ve sistem yöneticileri için yararlıdır.

Bytecode komutu sayesinde ek tespit mekanizmaları içeren bytecode.cvd dosyasını indirebilirsiniz. Bu direktif antivirüs tespit yeteneklerini arttırır. Ayrıca, bazı dosyaları güncelleme sürecinden muaf tutmamıza yarayan ***Exclude*** komutunu da kullanabilirsiniz.

Dosyaların izinlerini ise şu şekilde ayarlayabilirsiniz

```
root@samo:~ # ls -l /usr/local/bin/freshclam
-rwxr-xr-x  1 root wheel 58968 Sep 17 11:16 /usr/local/bin/freshclam
root@samo:~ # ls -l /usr/local/etc/freshclam.conf
-rw-r--r--  1 root wheel 7259 Oct 16 13:51 /usr/local/etc/freshclam.conf
root@samo:~ # ls -ld /var/log/clamav
drwxr-xr-x  2 clamav clamav 4 Sep 23 14:54 /var/log/clamav
root@samo:~ # ls -l /var/log/clamav/freshclam.log
-rw-r-----  1 clamav clamav 34518 Oct 16 13:08 /var/log/clamav/freshclam.log
root@samo:~ # ls -l /usr/local/etc/clamd.conf
-rw-r--r--  1 clamav clamav 28075 Sep 26 14:25 /usr/local/etc/clamd.conf
```

Loglarını kontrol etmek isterseniz ise aşağıdaki örneğe benzer bir çıktı görebilirsiniz :

```
root@samo:~ # tail -f /var/log/clamav/freshclam.log
daily.cld database is up-to-date (version: 27429, sigs: 2067241, f-level: 90, builder: raynman)
main.cvd database is up-to-date (version: 62, sigs: 6647427, f-level: 90, builder: sigmgr)
bytecode.cvd database is up-to-date (version: 335, sigs: 86, f-level: 90, builder: raynman)
-----------------------------------
Received signal: wake up
ClamAV update process started at Wed Oct 16 13:08:46 2024
daily.cld database is up-to-date (version: 27429, sigs: 2067241, f-level: 90, builder: raynman)
main.cvd database is up-to-date (version: 62, sigs: 6647427, f-level: 90, builder: sigmgr)
bytecode.cvd database is up-to-date (version: 335, sigs: 86, f-level: 90, builder: raynman)
-----------------------------------
```

Freshclam komutu ile ClamAV için güncellemeleri otomatik olarak kontrol edebilir ve indirilir.Şu şekilde bir çıktı verebilir:

```
root@samo:~ # freshclam
Wed Oct 16 14:09:34 2024 -> Current working dir is /var/db/clamav/
Wed Oct 16 14:09:34 2024 -> Loaded freshclam.dat:
Wed Oct 16 14:09:34 2024 ->   version:    1
Wed Oct 16 14:09:34 2024 ->   uuid:       fa65c6eb-bb81-4db1-a5e1-f4d1472e8783
Wed Oct 16 14:09:34 2024 -> ClamAV update process started at Wed Oct 16 14:09:34 2024
Wed Oct 16 14:09:34 2024 -> Current working dir is /var/db/clamav/
WARNING: Wed Oct 16 14:09:34 2024 -> DNS Update Info disabled. Falling back to HTTP mod
e.
Wed Oct 16 14:09:34 2024 -> Current working dir is /var/db/clamav/
Wed Oct 16 14:09:34 2024 -> check_for_new_database_version: Local copy of daily found:
daily.cld.
Wed Oct 16 14:09:34 2024 -> Trying to retrieve CVD header from http://mirror1.example.c
om/daily.cld
Wed Oct 16 14:09:34 2024 -> Local IPv4 address requested: 192.168.50.1
```

CLAMAV

ClamAV, Unix tabanlı işletim sistemlerinde kötü amaçlı yazılımları tespit etmek ve kaldırmak için kullanılan açık kaynaklı bir antivirüs yazılımıdır. Güvenlik odaklı bir çözüm sunan ClamAV, özellikle sunucu ortamlarında ve e-posta geçiş sistemlerinde yaygın olarak kullanılmaktadır. Kapsamlı bir virüs veritabanı ve etkili tarama mekanizmaları ile, sistem yöneticilerine ve kullanıcılarına güçlü bir savunma aracı sağlar.

Genişletilebilir Virüs Veritabanı: ClamAV, dünya genelindeki kullanıcılar tarafından düzenli olarak güncellenen bir virüs veritabanına sahiptir. Bu veritabanı, yeni kötü amaçlı yazılımlar, trojanlar, solucanlar ve diğer tehditleri içeren sürekli bir güncelleme döngüsüne sahiptir. Güncellemeler, freshclam aracıyla otomatik veya manuel olarak yapılabilir.

Çeşitli Tespit Yöntemleri: ClamAV, imza tabanlı tespit yönteminin yanı sıra heuristik ve davranışsal tespit yöntemlerini de kullanır. Bu, bilinmeyen tehditlere karşı bir güvenlik katmanı ekler. Özellikle heuristik analiz, şüpheli dosyaların davranışlarını inceleyerek daha proaktif bir koruma sunar.

E-posta Geçişi için Entegrasyon: ClamAV, e-posta sunucularıyla kolayca entegre edilebilir. Örneğin, Postfix veya Exim gibi popüler e-posta sunucuları ile birlikte kullanılarak, gelen ve giden e-postaların kötü amaçlı yazılımlara karşı taranmasını sağlar. Bu özellik, kurumsal e-posta güvenliğini artırır.

Komut Satırı Arayüzü: ClamAV, güçlü bir komut satırı arayüzü sunar. Bu, otomatik tarama, raporlama ve güncelleme süreçlerinin kolayca scriptlenmesine olanak tanır. Sistem yöneticileri, istedikleri ayarları özelleştirerek kendi ihtiyaçlarına göre yapılandırabilir.

Detaylı Loglama: ClamAV, tarama ve güncelleme işlemleriyle ilgili detaylı log kayıtları tutar. Bu loglar, sistem yöneticilerinin güvenlik tehditlerini izlemelerine ve analiz etmelerine yardımcı olur. Log dosyalarının boyutu, loglama düzeyi gibi ayarlar da özelleştirilebilir.

Hata Ayıklama ve Geliştirici Desteği: ClamAV, geliştiriciler için hata ayıklama mesajlarını etkinleştirme seçeneği sunar. Bu, sistem yöneticilerine ve geliştiricilere yazılımın davranışlarını analiz etme konusunda yardımcı olur.

Bytecode Tespiti: ClamAV, antivirüs tespit yeteneklerini artırmak için bytecode tabanlı tespit mekanizmaları içerir. Bu, daha karmaşık ve gelişmiş kötü amaçlı yazılımları tanımada yardımcı olur.

pkg install clamav komutu ile paketi indirebilirsiniz.

clamscan /path/to/directory komutu ile dosyaları veya dizinleri hedef alarak hedefli bir analiz yapılabilir

Clamd.conf , ClamAV'ın yapılandırma dosyasıdır.Clamd , ClamAV'ın daemon modunda çalışan bileşenidir ve genellikle dosyaların taranması için kullanılır. Bu dosya ise daemon'ın çalışma biçimini ve özelliklerini belirler

Clamd.conf dosyasında dikkat edilmesi gerekenler:

Oluşturulan logların nereye kaydedileceğini (clamd.log) belirtiniz. Bu sayede tarama işlemleri , algılanan tehditler veya hata mesajları hakkında bilgi alabilirsiniz

Log dosyalarının maksimum boyutunu ve her log mesajına zaman damgasını da ekleyebilirsiniz.(LogFileMaxSize , LogTime)

Pid dosyasını daha önce de açıklamıştık , bu dosyada da pid in path ini verebilirsiniz.

Clamav'ın virüs veritabanı dosyaları için de uygun dizini yoluyla beraber verebilirsiniz.(DatabaseDirectory)

Virüs bulunan dosyaların hash ve boyut gibi ek bilgilerini de loglayabilirsiniz. ExtendedDetectionInfo komutu sayesinde.

Ayrıntılı loglama yapmasını isterseniz ise LogVerbose direktifini etkinleştirebilirsiniz.

LogClean komutu sayesinde ise virüs bulaşmamış dosyaların da loglanmasını sağlayabilirsiniz.

Log dosyası belirli bir boyuta ulaştığında yenilenmesini LogRotate komutu sayesinde etkinleştirebilirsiniz.

ClamAV loglarının syslog ile hangi kategori altında kaydedileceğini belirleyebilirsiniz.

Tespit edilen zararlı dosyalar hakkında ek bilgi (dosya boyutu, hash değeri vb.) loglanmasını isterseniz ise ExtendedDetectionInfo direktifini etkinleştirebilirsiniz.

LocalSocketGroup komutuyla Unix socket dosyasının hangi grup tarafından sahipleneceğini belirtebilirsiniz.

ClamAV daemon düzgün kapatılmadığında stale socket dosyasını FixStaleSocket komutu ile kaldırabilirsiniz

TCPSocket komutu ile daemon'ın TCP/IP üzerinden belirli bir portu (3310) dinlemesini sağlayabilirsiniz.

MaxConnectionQueueLength direktifi ile bağlantı kuyruğundaki maksimum bekleyen bağlantı sayısını sınırlandırabilirsiniz.

Aynı anda çalışan maksimum iş parçacığı sayısını MaxThreads komutu ile ayarlayabilirsiniz.

Daemon'ın veri gönderme tamponu dolduğunda bekleyeceği maksimum süreyi (milisaniye cinsinden) SendBufTimeout direkitifi ile ayarlayabilirsiniz.

ExcludePath komutu , belirli bir yolun taranmasını engeller.

Tarama sırasında dizinlerin içine ne kadar derine inileceğini MaxDirectoryRecursion komutu ile belirleyebilirsiniz.

StreamMaxLength direktifi ile dosya taraması sırasında clamd'nin kabul edeceği maksimum veri büyüklüğünü belirleyebilirsiniz.

Belirli bir port aralığını sınırlandırarak güvenliği artırmak ve diğer süreçlerle port çatışmasını önlemek amacıyla StreamMinPort ve StreamMaxPort komutlarını kullanabilirsiniz

ClamAV'nin veritabanını periyodik olarak kontrol etme sıklığını SelfCheck direktifi sayesinde ayarlayabilirsiniz.

ConcurrentDatabaseReload komutu bir veritabanı yeniden yüklenirken ClamAV'nin tarama yapıp yapamayacağını kontrol eder. "No" ayarı, tarama yapılmasını engeller.

VirusEvent komutu sayesinde bir virüs tespit edildiğinde belirtilen komutun çalıştırılmasını sağlar. Burada, bir virüs bulunduğunda SMS ile bildirim gönderen bir komut tetiklenir.

ExitOnOOM komutu sayesinde ClamAV'nin bellek yetersizliği yaşandığında (Out of Memory - OOM) durdurulup durdurulmayacağını belirleyebilirsiniz.

Bu ve bunun gibi diğer direktifler ile daha gelişmiş ve ihtiyaçlarınıza yönelik şekilde kullanabilirsiniz

```
root@samo:~ # service clamav-clamd start
clamav_enable: YES -> YES
freeradius3_enable: YES -> YES
clamav_enable: YES -> YES
freeradius3_enable: YES -> YES
clamav_clamd already running?  (pid=28729).
root@samo:~ # ps aux | grep clamd
clamav  28729   0.0 19.1 1472308 1363172  -  Is   23:08      1:03.43 /usr/local/sbin/clamd
root    66336   0.0  0.0   12800    2516  0  S+   16:35      0:00.00 grep clamd
```

Log çıktısını kontrol etmek için ise bu örnekteki gibi yapabilirsiniz:

```
root@samo:~ # tail -f /var/log/clamav/clamd.log
Wed Oct 16 16:05:50 2024 -> Receive thread: closing conn (FD 11), group finished
Wed Oct 16 16:05:50 2024 -> Consumed entire command
Wed Oct 16 16:05:50 2024 -> Number of file descriptors polled: 1 fds
Wed Oct 16 16:05:50 2024 -> fds_poll_recv: timeout after 600 seconds
Wed Oct 16 16:15:50 2024 -> SelfCheck: Database status OK.
Wed Oct 16 16:15:50 2024 -> fds_poll_recv: timeout after 600 seconds
Wed Oct 16 16:25:50 2024 -> SelfCheck: Database status OK.
Wed Oct 16 16:25:50 2024 -> fds_poll_recv: timeout after 600 seconds
Wed Oct 16 16:35:50 2024 -> SelfCheck: Database status OK.
Wed Oct 16 16:35:50 2024 -> fds_poll_recv: timeout after 600 seconds
```

ClamAV Milter:

ClamAV antivirus yazılımıyla entegre bir Milter (Mail Filter) uygulamasıdır. Milter, bir e-posta sunucusu ile e-posta filtreleme sistemleri arasında bir köprü işlevi görerek, e-postaların güvenliğini sağlamak için kullanılır. ClamAV Milter, özellikle e-posta iletilerinin virüs taraması ve kötü amaçlı yazılımlara karşı korunması için tasarlanmıştır.

avantajları :

Gerçek Zamanlı Koruma: E-posta iletileri, sunucuya ulaşmadan önce taranarak, hızlı bir şekilde tehditlere karşı korunma sağlanır.

Açık Kaynak: ClamAV ve dolayısıyla ClamAV Milter, açık kaynaklıdır. Bu, maliyetleri düşürür ve kullanıcıların kaynak kodunu inceleyip özelleştirmesine olanak tanır.

Kolay Entegrasyon: ClamAV Milter, çeşitli e-posta sunucuları (Postfix, Sendmail gibi) ile kolayca entegre edilebilir. Bu, mevcut sistemlerin korunmasını sağlar.

Yüksek Esneklik: ClamAV Milter, yapılandırma dosyası aracılığıyla kullanıcı ihtiyaçlarına göre özelleştirilebilir. Kullanıcılar, hangi e-posta türlerinin taranacağını, karantinaya alınacağını veya kabul edileceğini belirleyebilir.

Detaylı Günlükleme: Tarama işlemleri ve sonuçları hakkında detaylı günlük kaydı tutarak, yöneticilerin sorun giderme süreçlerini kolaylaştırır.

!!

ClamAV ile beraber sisteminize gelip gelmediğini kontrol ediniz.

find / -name clamav-milter komutu ile bunu kontrol edebilirsiniz

Sisteminizde yüklü değilse

pkg install clamav-milter komutu ile paketi indirebilirsiniz.

Paketinizi yükledikten sonra ***clamav-milter.conf*** dosyasını açınız ve özelleştirmelerinizi yapınıız.

Yapılandırma dosyasında şunlara dikkat etmenizi tavsiye ederiz:

ClamAV Milter'ın bir Unix domain soketi üzerinden iletişim kuracağı dosya yolunu tanımlayabilirsiniz. Milter, mail sunucusu ile ClamAV arasında bir iletişim katmanı oluşturur. **MilterSocket** direktifi ile bu ayarı yapabilirsiniz.

MilterSocket inet:ip_adresiniz

: Bu komut, belirtilen IP adresi ve port üzerinden TCP soketi aracılığıyla iletişim kurulacağını belirtir. Bu iki **MilterSocket** tanımı, sistemin hem yerel Unix soketi hem de TCP/IP üzerinden iletişim kurmasına olanak tanır.

Unix soketinin erişim izinlerini belirleyebilirsiniz (tavsiye edilen 660). Bir önceki ClamAV Milter süreci düzgün şekilde kapatılmamışsa, kalıntı soket dosyasının silinip silinmeyeceğini **FixStaleSocket** direktifi ile belirleyebilirsiniz. Ayrıca, hangi kullanıcının altında çalışacağını da bu dosya içinde ayarlayabilirsiniz.

Süreç kimlik numarasının (PID) kaydedileceği dosya yolunu belirtebilirsiniz. Bu dosya, sürecin çalışıp çalışmadığını kontrol etmek için kullanılır.

ClamdSocket komutu ile ClamAV'ın tarama işlemleri için kullanacağı clamd ile iletişim kuracağı soketi tanımlayabilirsiniz. Burada **clamd.sock** dosyası kullanılır. Milter, bu soket aracılığıyla ClamAV'ın tarama işlemlerine erişir.

Belirttiğiniz IP adresinden gelen mesajları taramayabilirsiniz. Bunu **LocalNet** ile yapabilirsiniz; bu, performansınızı artırmak için faydalı olabilir. Ayrıca, taranacak e-postaların maksimum boyutunu **MaxFileSize** direktifi ile ayarlayabilirsiniz.

OnClean Accept: Tarama sonucunda zararsız bulunan e-postaların kabul edileceğini belirtir.

OnInfected Quarantine: Virüs bulunan e-postaların karantinaya alınacağını belirler. Yani, zararlı olarak tanımlanan e-postalar kullanıcıya ulaşmadan karantinaya gönderilir, bu da sistemi zararlı yazılımlardan korur.

Milter'ın çalışma durumunu ve tarama sonuçlarını kaydedeceği **LogFile** dosyasını tanımlayabilirsiniz. Günlük kayıtlarının daha ayrıntılı olması için **LogVerbose** komutunu da kullanabilirsiniz.

Dosyaların izinleri bu şekilde ayarlanılabilir.

```
root@samo:~ # ls -l /var/run/clamav/clmilter.sock
srwxrwxrwx  1 clamav clamav 0 Oct 16 16:05 /var/run/clamav/clmilter.sock
root@samo:~ # ls -l /var/run/clamav/clamav-milter.pid
-rw-r--r--  1 root wheel 6 Oct 16 16:05 /var/run/clamav/clamav-milter.pid
root@samo:~ # ls -l /var/run/clamav/clamd.sock
srw-rw-rw-  1 clamav clamav 0 Oct 11 15:37 /var/run/clamav/clamd.sock
root@samo:~ # ls -l /tmp/clamav-milter.log
-rw-r-----  1 clamav clamav 788 Oct 16 16:05 /tmp/clamav-milter.log
```

rc.conf un içerisine enable ettikten sonra ve rc.d klasörünün içeriisne gerekli bash komutunu girdikten sonra service komutu ile şu şekilde çalıştırabilirsiniz

```
root@samo:~ # service clamav-milter start
clamav_enable: YES -> YES
freeradius3_enable: YES -> YES
clamav_enable: YES -> YES
freeradius3_enable: YES -> YES
clamav_milter already running?  (pid=65356).
```

Konfigürasyon dosyasında belirttiğiniz path i ***tail -f /path/*** komutu ile girip şu şekilde bir çıktı ile karşılaşabilirsiniz:

```
root@samo:~ # tail -f /tmp/clamav-milter.log
Wed Oct 16 16:04:09 2024 -> +++ Started at Wed Oct 16 16:04:09 2024
Wed Oct 16 16:04:09 2024 -> Local socket unix:/var/run/clamav/clamd.sock added to the p
ool (slot 1)
Wed Oct 16 16:04:09 2024 -> Local socket unix:/var/run/clamav/clamd.sock added to the p
ool (slot 2)
Wed Oct 16 16:04:09 2024 -> Probe for slot 1 returned: success
Wed Oct 16 16:04:09 2024 -> Probe for slot 2 returned: success
Wed Oct 16 16:05:50 2024 -> +++ Started at Wed Oct 16 16:05:50 2024
Wed Oct 16 16:05:50 2024 -> Local socket unix:/var/run/clamav/clamd.sock added to the p
ool (slot 1)
Wed Oct 16 16:05:50 2024 -> Local socket unix:/var/run/clamav/clamd.sock added to the p
ool (slot 2)
Wed Oct 16 16:05:50 2024 -> Probe for slot 1 returned: success
Wed Oct 16 16:05:50 2024 -> Probe for slot 2 returned: success
```

SURİCATA : Gelişmiş Ağ Tehdit Algılama Motoru

Suricata, ücretsiz ve açık kaynak kodlu, olgun, hızlı ve sağlam bir ağ tehdit algılama motorudur. Gelişmiş güvenlik özellikleri sunan bu sistem, gerçek zamanlı saldırı algılama (IDS), satır içi saldırı önleme (IPS), ağ güvenliği izleme (NSM) ve çevrimdışı PCAP işleme yeteneklerine sahiptir. Suricata, ağ trafiğini incelemek için güçlü ve kapsamlı bir kurallar ve imza dili kullanırken, karmaşık tehditlerin tespiti için Lua komut dosyası desteği sunar.

Temel Özellikler

Geniş Protokol Desteği: Suricata, TCP, UDP, ICMP gibi temel ağ protokollerinin yanı sıra HTTP, DNS, FTP ve diğer üst düzey protokolleri de destekler. Bu, çeşitli ağ trafiği türlerinin detaylı analizi için olanak tanır.

Hızlı ve Verimli İşlem: Suricata, çoklu iş parçacığı desteği sayesinde çok çekirdekli işlemcilerden yararlanarak yüksek hacimli ağ trafiğini etkili bir şekilde işleyebilir. Bu, gerçek zamanlı saldırı algılama ve önleme süreçlerini hızlandırır.

Kurallar ve İmzalar: Suricata, kullanıcıların özel kurallar oluşturmasına ve bunları mevcut kural setleriyle birleştirmesine olanak tanır. Snort kural setleri ile uyumluluğu sayesinde, Snort kullanıcıları kolaylıkla Suricata'ya geçiş yapabilir.

YAML ve JSON Entegrasyonu: Suricata, mevcut SIEM çözümleri (Splunk, Logstash/Elasticsearch, Kibana gibi) ile kolayca entegre olabilir. Bu, verilerin analizi ve raporlanması sürecini zahmetsiz hale getirir.

Anomali Tespiti: Ağ trafiğinde anomali tespiti yaparak olağandışı davranışları belirleyebilir. Bu özellik, sadece saldırıları değil, aynı zamanda ağdaki potansiyel sorunları da tespit etmeye yardımcı olur.

Gelişmiş Özellikler

a. **Hızlı Algılama Motoru:**

Suricata, hem signature (imza) hem de anomali tespiti için hızlı bir algılama motoru kullanır. Bu motor, ağ trafiğini gerçek zamanlı olarak analiz edebilir ve belirli davranışlara dayalı olarak hızlı yanıt verir.

b. **PCAP Desteği:**

Suricata, ağ trafiğini PCAP formatında kaydedebilir. Bu özellik, geçmişteki ağ trafiğini analiz etmek için kullanılabilir ve olay sonrası analiz için büyük bir avantaj sağlar.

c. **SSL/TLS İncelemesi:**

Suricata, SSL ve TLS trafiğini de analiz edebilir. Bu, şifreli ağ trafiği içindeki potansiyel tehditlerin tespit edilmesine yardımcı olur. HTTPS trafiğinin içeriğini incelemek için SSL/TLS sertifikalarını yönetme yeteneğine sahiptir.

d. **DDoS Koruması:**

Suricata, dağıtık hizmet reddi (DDoS) saldırılarını tespit etmek ve önlemek için çeşitli mekanizmalar içerir. Bu tür saldırılara karşı koruma sağlamak, ağ güvenliğini artırır.

e. **Entegre Ağ Yük Dengeleyicisi:**

Suricata, yük dengeleme ve yönlendirme gibi özellikler de sunarak, ağ trafiğini optimize eder. Bu, daha iyi bir ağ performansı ve güvenliği sağlar.

Kullanım Alanları

Suricata, birçok farklı kullanım senaryosunda etkilidir:

Ağ Güvenliği: Ağ trafiğini izleyerek potansiyel tehditleri tespit eder ve güvenlik ihlallerine karşı koruma sağlar.

Olay Yönetimi: Güvenlik olaylarını kaydederek, olay sonrası analiz ve raporlama için kullanılabilir.

Ağ Analizi: Ağ performansını izleyerek potansiyel sorunları belirler ve optimize eder.

Kurulum adımları

portsnap fetch update

pkg install suricata

pkg install oinkmaster

ee /usr/local/etc/oinkmaster.conf

url = http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz

have not researched enough yet

url = https://www.snort.org/rules/community

requires registration

 # url = https://www.snort.org/rules/snortrules-snapshot-DDDD.tar.gz?oinkcode=<oinkcode>

crontab -e

*51 */12 * * * /usr/local/sbin/oinkmaster -Q -C /usr/local/etc/oinkmaster.conf -o /usr/local/etc/suricata/rules*

nano /usr/local/bin/run-oinkmaster.sh

#!/bin/sh

cd /usr/local/etc

/usr/sbin/oinkmaster -Q -C /usr/local/etc/oinkmaster.conf -o /usr/local/etc/suricata/rules

sleep 5

kill -USR2 `cat /var/run/suricata.pid`

chmod +x /usr/local/bin/run-oinkmaster.sh

mkdir -p /var/lib/suricata/rules

nano /usr/local/etc/oinkmaster.conf

output_dir = /var/lib/suricata/rules

nano /usr/local/etc/suricata/suricata.yaml

rule-files:

- /var/lib/suricata/rules/app-layer-events.rules

- /var/lib/suricata/rules/botcc.portgrouped.rules

- /var/lib/suricata/rules/botcc.rules

- /var/lib/suricata/rules/ciarmy.rules

- /var/lib/suricata/rules/compromised.rules

- /var/lib/suricata/rules/decoder-events.rules

- /var/lib/suricata/rules/dns-events.rules

- /var/lib/suricata/rules/drop.rules

- /var/lib/suricata/rules/dshield.rules

- /var/lib/suricata/rules/emerging-activex.rules

- /var/lib/suricata/rules/emerging-attack_response.rules

- /var/lib/suricata/rules/emerging-current_events.rules

- /var/lib/suricata/rules/emerging-dns.rules

/usr/sbin/oinkmaster -Q -C /usr/local/etc/oinkmaster.conf -o /var/lib/suricata/rules

chown -R suricata:suricata /var/lib/suricata/rules

chmod -R 750 /var/lib/suricata/rules

ls -ld /var/lib/suricata/rules

drwxr-x--- 2 suricata suricata 512 Oct 17 12:34 /var/lib/suricata/rules

service suricata restart

!!!

Olası hataları tail -f /var/log/messages ile bakabilirsiniz

suricata.yaml dosyası, Suricata IDS/IPS (Intrusion Detection System/Intrusion Prevention System) yazılımının yapılandırma dosyasıdır. Bu dosya, Suricata'nın çalışma şekli ve analiz edeceği ağ trafiği için çeşitli ayarları içerir. İşte bu dosyanın bazı ana bileşenleri ve işlevleri:

1.Ağ Arayüzleri (Network Interfaces)

Suricata'nın hangi ağ arayüzlerini dinleyeceğini belirtir. Genellikle bir veya daha fazla ağ arayüzü tanımlanır.

2. Kural Yönetimi (Rule Management)

Suricata, belirli olayları algılamak için kural dosyalarını kullanır. suricata.yaml dosyasında, bu kural dosyalarının yolları ve kuralların nasıl yükleneceği ile ilgili ayarlar bulunur.

3. Günlük Ayarları (Logging Settings)

Suricata'nın günlükleme (logging) ayarları, günlüklerin hangi formatta kaydedileceği, nereye kaydedileceği ve hangi seviyede günlükleme yapılacağı gibi bilgileri içerir.

4. Performans ve Bellek Ayarları

Suricata'nın performansını etkileyen ayarlar, örneğin kaç iş parçacığı (thread) kullanılacağı ve bellek yönetimi ile ilgili parametreler burada tanımlanır.

5. Güvenlik ve Politika Ayarları

Ağ trafiği ile ilgili güvenlik politikalarının ve filtreleme kurallarının ayarlandığı bölümler bulunur. Hangi tür trafiğin izleneceği veya engelleneceği gibi bilgiler içerir.

6. Protokol Analizi

Suricata, çeşitli ağ protokollerini analiz edebilir. Bu ayarlar, hangi protokollerin izleneceği ve analiz edileceği ile ilgilidir.

7. Özelleştirilmiş Modüller

Suricata, çeşitli modülleri ve eklentileri destekler. suricata.yaml dosyasında bu modüllerin ayarları ve etkinleştirilmesi ile ilgili bilgiler bulunabilir.

8. Altyapı Ayarları

Suricata'nın çalışma ortamı ile ilgili ayarlar, örneğin ağ trafiğinin yönlendirilmesi veya yeniden yönlendirilmesi gibi yapılandırmalar burada yapılabilir.

9. Yapılandırma Örnekleri ve Belgeler

Suricata'nın resmi belgelerine ve yapılandırma örneklerine bağlantılar içerir, böylece kullanıcılar için referans sağlar.

Suricata.yaml dosyasının içeriği:

Vars (Değişkenler)

Bu bölümdeki değişkenler, ağ yapılandırmasını daha okunabilir hale getirmek için gruplar halinde düzenlenmiştir.

Address Groups (Adres Grupları)

HOME_NET: Yerel ağ adreslerini tanımlar. Örneğin, 192.168.10.0/24 yerel ağınızın adres aralığını belirtir.

EXTERNAL_NET: Dış ağı tanımlar ve !$HOME_NET ifadesi ile yerel ağ dışındaki tüm adresleri belirtir.

HTTP_SERVERS: HTTP sunucularının adreslerini belirtir, bu durumda yerel ağdaki sunucular kullanılır.

SMTP_SERVERS: SMTP (Mail) sunucularını belirtir, yerel ağ içindeki sunucular.

SQL_SERVERS: SQL sunucularının adreslerini tanımlar, yine yerel ağda bulunur.

DNS_SERVERS: DNS sunucularının adreslerini belirtir, yerel ağdaki sunucular.

TELNET_SERVERS: Telnet sunucularını tanımlar, yerel ağdaki sunucular.

AIM_SERVERS: Dış ağdaki AIM (AOL Instant Messenger) sunucularını belirtir.

DC_SERVERS: Yerel ağda bulunan DC (Direct Connect) sunucularını tanımlar.

DNP3_SERVER: DNP3 protokolünü kullanan sunucuların adreslerini tanımlar, yerel ağda.

DNP3_CLIENT: DNP3 protokolünü kullanan istemcilerin adreslerini tanımlar, yerel ağda.

MODBUS_CLIENT: Modbus protokolü kullanan istemcilerin adreslerini tanımlar, yerel ağda.

MODBUS_SERVER: Modbus protokolü kullanan sunucuların adreslerini tanımlar, yerel ağda.

ENIP_CLIENT: ENIP (Ethernet/IP) protokolünü kullanan istemcilerin adreslerini belirtir, yerel ağda.

ENIP_SERVER: ENIP protokolünü kullanan sunucuların adreslerini belirtir, yerel ağda.

Port Groups (Port Grupları)

HTTP_PORTS: HTTP trafiği için kullanılan portları tanımlar (varsayılan olarak 80).

SHELLCODE_PORTS: Shellcode (zararlı kod) için kullanılan portları tanımlar. !80 ifadesi, port 80 dışındaki tüm portları içerir.

ORACLE_PORTS: Oracle veritabanı sunucuları için kullanılan port (1521).

SSH_PORTS: SSH (Secure Shell) protokolü için kullanılan port (22).

DNP3_PORTS: DNP3 protokolü için kullanılan port (20000).

MODBUS_PORTS: Modbus protokolü için kullanılan port (502).

FILE_DATA_PORTS: HTTP ve diğer ilgili portların kombinasyonu (80, 110, 143).

FTP_PORTS: FTP (File Transfer Protocol) için kullanılan port (21).

GENEVE_PORTS: Geneve protokolü için kullanılan port (6081).

VXLAN_PORTS: VXLAN (Virtual Extensible LAN) için kullanılan port (4789).

TEREDO_PORTS: Teredo tünelleme protokolü için kullanılan port (3544).

Default Log Directory

default-log-dir: Suricata'nın günlük dosyalarının saklanacağı dizin. Varsayılan olarak /var/log/suricata/ olarak ayarlanmıştır.

Default Log Directory

default-log-dir: Suricata'nın günlük dosyalarının saklanacağı dizin. Varsayılan olarak /var/log/suricata/ olarak ayarlanmıştır.

Plugins (Eklentiler)

Bu bölümde, Suricata'nın kullanılacak eklentilerini tanımlamak için bir yapı sağlanır.

plugins: Eklentiler için yapılandırma bölümü. Örnek eklenti yolu yorum satırı olarak bırakılmıştır.

Outputs (Çıktılar)

Bu bölüm, Suricata'nın hangi tür çıktı dosyalarını üreteceğini belirtir.

fast: Hızlı günlüklere yönelik yapılandırma.

enabled: Fast günlüğün etkin olup olmadığını belirtir (yes veya no).

filename: Fast günlüğün dosya adı (örneğin, fast.log).

append: Günlüğün mevcut dosyaya eklenip eklenmeyeceğini belirtir (yes veya no).

eve-log: Eve formatında günlüklerin yapılandırması.

enabled: Eve günlüklerinin etkin olup olmadığını belirtir.

filetype: Günlük dosya türü (örneğin, regular).

filename: Eve günlüğünün dosya adı (örneğin, eve.json).

community-id: Community ID'nin etkin olup olmadığını belirtir.

xff: X-Forwarded-For başlığı ile ilgili ayarlar.

enabled: XFF'nin etkin olup olmadığını belirtir.

mode: XFF'nin nasıl kullanılacağını belirtir.

deployment: Kullanım türünü belirtir.

header: Kullanılacak başlık ismi.

types: Farklı güncel türleri tanımlar:

alert: Uyarı günlüğü.

tagged-packets: Etiketli paketlerin günlüklenip günlüklenmeyeceği.

anomaly: Anomalileri günlüğe kaydetme.

enabled: Anomalilerin etkin olup olmadığını belirtir.

http: HTTP günlüğü.

extended: Genişletilmiş günlükleme.

tls: TLS günlükleme.

extended: Genişletilmiş günlükleme.

files: Dosyalarla ilgili günlükleme.

force-magic: Dosya türünün zorlanıp zorlanmayacağı.

smtp: SMTP günlükleme.

file-store: Dosya depolama için yapılandırma.

version: Versiyon numarası (2).

enabled: Dosya depolamanın etkin olup olmadığını belirtir.

xff: XFF ayarları (daha önce açıklandığı gibi).

tcp-data: TCP verisi günlükleme yapılandırması.

enabled: TCP verisinin etkin olup olmadığını belirtir.

type: Günlük türü (file).

filename: TCP verisinin günlük dosyası adı.

http-body-data: HTTP gövde verisi günlükleme yapılandırması.

enabled: HTTP gövde verisinin etkin olup olmadığını belirtir.

type: Günlük türü (file).

filename: HTTP gövde verisi dosya adı.

lua: Lua eklentileri için yapılandırma.

enabled: Lua eklentilerinin etkin olup olmadığını belirtir.

scripts: Lua scriptlerinin listesi.

Logging (Günlükleme)

Bu bölüm, günlükleme ayarlarını yapılandırmak için kullanılır.

default-log-level: Varsayılan günlükleme seviyesi (örneğin, notice).

outputs: Günlükleme çıktılarının yapılandırılması.

console: Konsol çıktısını belirtir.

enabled: Konsol günlüğünün etkin olup olmadığını belirtir.

file: Dosyaya günlük kaydını belirtir.

enabled: Dosya günlüğünün etkin olup olmadığını belirtir.

level: Günlükleme seviyesi (örneğin, info).

filename: Günlük dosyası adı (suricata.log).

syslog: Syslog çıktısını belirtir.

enabled: Syslog günlüğünün etkin olup olmadığını belirtir.

facility: Syslog tesisi (örneğin, local5).

format: Günlük formatı.

AF Packet (AF Paketi)

Bu bölüm, ağ paketlerini işlemek için kullanılan yapılandırmadır.

af-packet: Ağ arabirimleri için yapılandırma.

interface: Kullanılacak ağ arabirimi (örneğin, eth0).

cluster-id: Küme kimliği.

cluster-type: Küme türü (none).

Network (Ağ)

Bu bölüm, ağ yapılandırması ile ilgili genel ayarları içerir.

network: Ağ yapılandırması.

type: Ağ tipi (ethernet).

promisc: Promiscuous modun etkin olup olmadığını belirtir.

PCAP (Paket Yakalama)

Bu bölüm, paket yakalama yapılandırmasını içerir.

pcap

interface: Paketlerin yakalanacağı ağ arayüzlerini belirtir.

eth0: Belirtilen bir ağ arayüzü.

default: Varsayılan ağ arayüzü.

PCAP Dosyası

Bu bölüm, PCAP dosyası ile ilgili yapılandırma seçeneklerini içerir.

pcap-file

checksum-checks: Paketlerin kontrol toplamı denetimlerinin nasıl yapılacağını belirler.

auto: Kontrol toplamı denetimleri otomatik olarak yapılır.

Uygulama Katmanı Protokolleri

Bu bölüm, uygulama katmanı protokollerinin yapılandırmasını içerir. Her bir protokol, güvenlik izleme ve tespit için yapılandırılabilir.

app-layer

protocols: İzlenmek istenen uygulama katmanı protokollerinin listesi.

telnet:

enabled: Telnet protokolü etkin.

rfb (Remote Framebuffer):

enabled: RFB etkin.

detection-ports: İzleme yapılacak bağlantı noktaları.

dp: 5900 ile 5909 arasındaki portlar.

mqtt:

enabled: MQTT protokolü etkin.

krb5:

enabled: Kerberos 5 etkin.

bittorrent-dht:

enabled: BitTorrent DHT etkin.

snmp:

enabled: SNMP etkin.

ike:

enabled: IKE (Internet Key Exchange) etkin.

tls:

enabled: TLS etkin.

detection-ports: İzleme yapılacak bağlantı noktası.

dp: 443 portu.

pgsql:

enabled: PostgreSQL etkin değil.

stream-depth: Akış derinliği (0).

dcerpc:

enabled: DCERPC etkin.

ftp:

enabled: FTP etkin.

rdp:

enabled: RDP etkin değil (yorum satırı).

ssh:

enabled: SSH etkin.

http2:

enabled: HTTP/2 etkin.

smtp:

enabled: SMTP etkin.

raw-extraction: Ham verinin çıkarılıp çıkarılmayacağını belirtir (no).

mime: MIME ayarları.

decode-mime: MIME'nin çözülüp çözülemeyeceği (yes).

decode-base64: Base64 çözümlemesi (yes).

decode-quoted-printable: Quoted-printable çözümlemesi (yes).

header-value-depth: Başlık değeri derinliği (2000).

extract-urls: URL'lerin çıkarılıp çıkarılmayacağı (yes).

body-md5: Gövdenin MD5 kontrol toplamı çıkarılacak mı (no).

imap:

enabled: Sadece tespit için etkin.

smb:

enabled: SMB etkin.

detection-ports: İzlenecek bağlantı noktaları.

dp: 139, 445 portları.

nfs:

enabled: NFS etkin.

tftp:

enabled: TFTP etkin.

tcp:

enabled: TCP üzerinden DNS etkin.

detection-ports: İzleme yapılacak bağlantı noktası.

dp: 53 portu.

udp:

enabled: UDP üzerinden DNS etkin.

detection-ports: İzleme yapılacak bağlantı noktası.

dp: 53 portu.

http:

enabled: HTTP etkin.

LibHTP Ayarları

Bu bölüm, LibHTP yapılandırmasını içerir. HTTP analizinde kullanılan ayarları belirler.

libhtp

default-config: Varsayılan yapılandırma ayarları.

personality: Kullanıcı deneyimi (IDS).

request-body-limit: İstek gövdesinin maksimum boyutu (100kb).

response-body-limit: Cevap gövdesinin maksimum boyutu (100kb).

request-body-minimal-inspect-size: Minimum inceleme boyutu (32kb).

request-body-inspect-window: İnceleme penceresi boyutu (4kb).

response-body-minimal-inspect-size: Cevap gövdesi için minimum inceleme boyutu (40kb).

response-body-inspect-window: Cevap gövdesinin inceleme penceresi boyutu (16kb).

response-body-decompress-layer-limit: Cevap gövdesi için maksimum dekompresyon katmanı sayısı (2).

http-body-inline: HTTP gövdesinin çevrimiçi izlenip izlenmeyeceğini belirtir (auto).

swf-decompression: SWF dosyalarının dekompresyon ayarları.

enabled: SWF dekompresyonunun etkin olup olmadığını belirtir (no).

type: Kullanılacak dekompresyon türü (both).

compress-depth: Maksimum sıkıştırma derinliği (100kb).

decompress-depth: Maksimum dekompresyon derinliği (100kb).

double-decode-path: Çift kod çözme yolu (no).

double-decode-query: Çift kod çözme sorgusu (no).

server-config

Uncomment and modify: İhtiyaçlarınıza göre özel sunucu yapılandırmalarını açıp değiştirebilirsiniz.

apache:

address: Apache sunucusunun kabul ettiği IP adres aralıkları.

[192.168.1.0/24, 127.0.0.0/8, "::1"]: IPv4 ve IPv6 adresleri.

personality: Sunucu türü (Apache_2).

request-body-limit: İstek gövdesinin maksimum boyutu (4096 bayt).

response-body-limit: Cevap gövdesinin maksimum boyutu (4096 bayt).

iis7:

address: IIS sunucusunun kabul ettiği IP adres aralıkları.

192.168.0.0/24: IP aralığı.

192.168.10.0/24: IP aralığı.

personality: Sunucu türü (IIS_7_0).

request-body-limit: İstek gövdesinin maksimum boyutu (4096 bayt).

response-body-limit: Cevap gövdesinin maksimum boyutu (4096 bayt).

double-decode-path: Çift kod çözme yolu (no).

double-decode-query: Çift kod çözme sorgusu (no).

Modbus

Bu bölüm, Modbus protokolü ile ilgili yapılandırmayı içerir.

modbus

enabled: Modbus etkin değil (no).

detection-ports: İzleme yapılacak bağlantı noktası.

dp: 502 portu.

stream-depth: Akış derinliği .

DNP3

Bu bölüm, DNP3 protokolü ile ilgili yapılandırmayı içerir.

dnp3

enabled: DNP3 etkin değil (no).

detection-ports: İzleme yapılacak bağlantı noktası.

dp: 20000 portu.

ENIP

Bu bölüm, ENIP (Ethernet/IP) protokolü ile ilgili yapılandırmayı içerir.

enip

enabled: ENIP etkin değil (no).

detection-ports: İzleme yapılacak bağlantı noktaları.

dp: 44818 portu.

sp: 44818 portu.

NTP

Bu bölüm, NTP (Network Time Protocol) ile ilgili yapılandırmayı içerir.

ntp

enabled: NTP etkin (yes).

QUIC

Bu bölüm, QUIC (Quick UDP Internet Connections) protokolü ile ilgili yapılandırmayı içerir.

quic

enabled: QUIC etkin (yes).

DHCP

Bu bölüm, DHCP (Dynamic Host Configuration Protocol) ile ilgili yapılandırmayı içerir.

dhcp

enabled: DHCP etkin (yes).

SIP

Bu bölüm, SIP (Session Initiation Protocol) ile ilgili yapılandırmayı içerir.

security

limit-noproc: Süreç sınırlandırması (true).

landlock: Landlock güvenlik özelliği.

enabled: Landlock etkin değil (no).

directories: Okunabilecek dizinler.

read: Okunabilecek dizinler listesi.

/usr/

/etc/

/usr/local/etc/suricata/

lua: Lua betik desteği.

allow-rules: Kural izinleri (false).

Coredump Ayarları

Bu bölüm, çekirdek dökümü yapılandırmasını içerir.

coredump

max-dump: Maksimum çekirdek döküm boyutu (unlimited).

Genel Çalışma Ayarları

Bu bölüm, genel çalışma ayarlarını içerir.

host-mode

host-mode: Ana bilgisayar modu (auto).

unix-command

enabled: Unix komutu desteği (auto).

filename: Özel soket dosyası (yorum satırı).

Legacy Ayarları

Bu bölüm, eski yapılandırma ayarlarını içerir.

legacy

uricontent: URI içeriği desteği (enabled).

Tespit Ayarları

Bu bölüm, tespit ile ilgili yapılandırma ayarlarını içerir.

Detection settings

action-order: Eylem sırası (yorum satırı).

packet-alert-max: Maksimum paket uyarısı sayısı (yorum satırı).

exception-policy: İstisna politikası (auto).

Diğer Ayarlar

engine-analysis: Motor analizi ayarları.

rules-fast-pattern: Hızlı desen kuralları etkin (yes).

rules: Tüm kurallar etkin (yes).

PCRE (Perl Compatible Regular Expressions)

match-limit: Eşleşmelerin sayısını sınırlamak için bir üst sınır belirler. Burada, maksimum 3500 eşleşmeye izin veriliyor.

match-limit-recursion: Regüler ifadelerin iç içe geçişlerinde en fazla 1500 kez eşleşme yapılmasına izin verilir.

Host OS Policy

Bu bölüm, belirli işletim sistemlerine göre kurallar tanımlar.

windows: Windows işletim sistemleri için tüm IP adreslerine (0.0.0.0/0) izin verilir. Diğer işletim sistemleri için herhangi bir kural tanımlanmamış.

Defrag (Parçaları Birleştirme)

Bu bölüm, ağ trafiği analizinde kullanılan fragmentlerin yönetimi ile ilgilidir.

memcap: 32 MB bellek ayırır.

hash-size: Fragmentler için kullanılan hash tablosunun boyutunu belirtir (65536).

trackers: İzleme için kullanılan maksimum sayıyı belirler (65535).

max-frags: Yönetilebilecek maksimum fragment sayısını belirler (65535).

prealloc: Belleğin önceden tahsis edilip edilmeyeceğini belirler (evet).

timeout: Fragmentlerin zaman aşımını 60 saniye olarak ayarlar.

Flow (Akış)

Ağ akışlarını yönetmek için yapılandırma ayarlarıdır.

memcap: 128 MB bellek ayırır.

hash-size: Akışlar için kullanılan hash tablosunun boyutunu belirtir (65536).

prealloc: Belleğin önceden tahsis edilip edilmeyeceğini belirtir (10000).

emergency-recovery: Acil durum geri kazanım süresini belirtir (30 saniye).

VLAN

use-for-tracking: VLAN bilgilerini izlemek için kullanılacağını belirtir (doğru).

Live Development

use-for-tracking: Canlı geliştirme bilgilerini izlemek için kullanılacağını belirtir (doğru).

Flow Timeouts

Ağ trafiği için zaman aşımı ayarlarıdır.

default: Tüm yeni bağlantılar için zaman aşımı süresi (30 saniye) ve durumları için farklı zaman aşımı süreleri tanımlar.

tcp: TCP bağlantıları için ayrı zaman aşımı değerleri (yeni, kurulu, kapalı).

udp: UDP bağlantıları için zaman aşımı süreleri.

icmp: ICMP bağlantıları için zaman aşımı süreleri.

Stream (Akış)

memcap: 64 MB bellek ayırır.

checksum-validation: Kontrol toplamı doğrulamasının etkin olup olmadığını belirtir (evet).

inline: Akış yönetim modunu otomatik olarak ayarlar.

reassembly: Parçaları birleştirmek için kullanılan ayarlar. Bellek, derinlik ve parça boyutları gibi ayarlar içerir.

Host

hash-size: Host bilgileri için hash tablosunun boyutu (4096).

prealloc: Önceden tahsis edilen host sayısı (1000).

memcap: 32 MB bellek ayırır.

Decoder (Ayrıştırıcı)

teredo: Teredo protokolünün etkin olup olmadığını belirtir ve hangi portların kullanılacağını ayarlar.

vxlan: VXLAN protokolünün etkin olup olmadığını belirtir ve hangi portların kullanılacağını ayarlar.

Threading (İş Parçacığı)

set-cpu-affinity: CPU affinitesi ayarının etkin olup olmadığını belirtir (hayır).

cpu-affinity: Farklı CPU setleri için yönetim, alma ve işçi görevleri için ayarlar içerir.

detect-thread-ratio: İş parçacığı algılama oranını belirtir (1.0).

luajit: LuaJIT için durum sayısını belirtir (128).

Profiling (Profil Oluşturma)

Ağ trafiği ve kuralların performansını izlemek için ayarlar içerir.

rules, keywords, prefilter, rulegroups, packets, locks, pcap-log: Her biri için ayrı ayrı günlük dosyası ayarları bulunur (örneğin, günlük dosyası adı, ekleme durumu, sınır).

NFQ ve NFLOG

Bu bölümler, Netfilter Queue ve Netfilter Logging için ayarları içerir. Örneğin, grup ve tampon boyutu gibi parametreler.

Netmap

interface: Kullanılacak ağ arayüzlerini belirtir (örneğin, eth2 ve default).

/your/to/path/suricata/reference.config

Referans klasorüdür.

Bugtraq:

URL: http://www.securityfocus.com/bid/

Açıklama: Güvenlik açıkları ile ilgili bilgiler sağlayan bir veri tabanıdır. Suricata, Bugtraq referanslarını kullanarak belirli tehditleri ve zafiyetleri tanımlamak için kullanılabilir.

BID:

URL: http://www.securityfocus.com/bid/

Açıklama: Bugtraq ile aynı amaçla hizmet eden bir başka referans noktasıdır. Zafiyet bilgileri ve ilgili düzeltme notlarını içerir.

CVE:

URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=

Açıklama: Ortak Zafiyet ve Maruz Kalma Listesi'dir (Common Vulnerabilities and Exposures). Suricata, bu referansı kullanarak bilinen zafiyetlere karşı ağ trafiğini analiz edebilir.

Secunia:

URL: http://www.secunia.com/advisories/

Açıklama: Güvenlik açıkları ve yazılım güncellemeleri hakkında bilgi sağlayan bir platformdur. Suricata, bu bilgileri kullanarak güncel tehditlere karşı koruma sağlayabilir.

ArachNIDS:

URL: http://www.whitehats.com/info/IDS

Açıklama: İnternet üzerindeki güvenlik açıkları ve tehditler hakkında bilgi sağlayan bir kaynak. Artık mevcut değil ama önceki bilgiler hala Suricata'nın güvenlik analizi için referans olarak kullanılabilir.

McAfee:

URL: http://vil.nai.com/vil/content/v_

Açıklama: McAfee'nin güvenlik tehditleri ve virüsleri hakkında bilgi sağlayan bir veri tabanıdır. Suricata, bu bilgileri tehdit analizi için kullanabilir.

Nessus:

URL: http://cgi.nessus.org/plugins/dump.php3?id=

Açıklama: Ağ güvenlik açıklarını taramak için kullanılan bir araçtır. Suricata, Nessus ile entegre edilerek daha etkili bir güvenlik analizi gerçekleştirebilir.

ET (Emerging Threats):

URL: http://doc.emergingthreats.net/

Açıklama: Yeni ve gelişen tehditlere karşı kural ve veri sağlayan bir kaynaktır. Suricata, bu kuralları kullanarak tehditleri tespit etme yeteneğini artırabilir.

OSVDB:

URL: http://osvdb.org/show/osvdb/

Açıklama: Açık kaynaklı zafiyet veri tabanı. Suricata, OSVDB'den alınan bilgilerle tehditleri daha iyi değerlendirebilir.

ThreatExpert:

URL: http://www.threatexpert.com/report.aspx?md5=

Açıklama: Kötü amaçlı yazılımların analizi hakkında bilgi sağlayan bir platformdur. Suricata, bu bilgileri kullanarak kötü amaçlı yazılımlara karşı koruma sağlayabilir.

ExploitDB:

URL: http://www.exploit-db.com/exploits/

Açıklama: Bilinen güvenlik açıkları için açık kaynaklı bir veri tabanı. Suricata, burada listelenen açıkları tespit etmek için kullanılabilir.

OpenPacket:

URL: https://www.openpacket.org/capture/grab/

Açıklama: Ağ paketlerinin yakalanması ve analizi için bir kaynak. Suricata, bu verileri kullanarak ağ trafiğini analiz edebilir.

SecurityTracker:

URL: http://securitytracker.com/id?

Açıklama: Güvenlik açıkları ve tehditlerle ilgili güncel bilgiler sağlayan bir platformdur. Suricata, bu bilgileri kullanarak ağ güvenliğini artırabilir.

X-Force:

URL: http://xforce.iss.net/xforce/xfdb/

Açıklama: IBM'in güvenlik araştırma ekibi tarafından sağlanan bilgi kaynağıdır. Suricata, X-Force verileri ile bilinen tehditleri tanımlamak için kullanılabilir.

Microsoft:

URL: http://technet.microsoft.com/security/bulletin/

Açıklama: Microsoft'un güvenlik güncellemeleri ve zafiyetleri hakkında bilgi sağladığı resmi kaynaktır. Suricata, bu bilgilerle Microsoft ürünlerindeki güvenlik açıklarını analiz edebilir.

!!!

Rules dosyalarında dikkat edilmesi gereken bir diğer önemli nokta ise her bir kuralın birbirinden faklı pid değerlerini sahip olmasıdır.

```
root@samo:~ # service suricata start
clamav_enable: YES -> YES
freeradius3_enable: YES -> YES
clamav_enable: YES -> YES
freeradius3_enable: YES -> YES
suricata already running?  (pid=29959).
root@samo:~ # service suricata status
clamav_enable: YES -> YES
freeradius3_enable: YES -> YES
clamav_enable: YES -> YES
freeradius3_enable: YES -> YES
suricata is running as pid 29959.
```

!!

Ekranı büyültmeniz halinde detaylı bilgilere erişeceksiniz.

```
root@samo:~ # tail -f /var/log/suricata/fast.log
10/17/2024-14:49:16.836422  [**] [1:2231000:1] SURICATA QUIC failed decrypt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.10.158:53734 -> 142.250.180.13
2:443
10/17/2024-14:49:23.841821  [**] [1:2231000:1] SURICATA QUIC failed decrypt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.10.158:58727 -> 216.58.204.227
:443
10/17/2024-14:49:23.917978  [**] [1:2231000:1] SURICATA QUIC failed decrypt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.10.158:58727 -> 216.58.204.227
:443
10/17/2024-14:49:29.670203  [**] [1:2231000:1] SURICATA QUIC failed decrypt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.10.158:56414 -> 142.250.180.13
2:443
10/17/2024-14:49:29.670407  [**] [1:2231000:1] SURICATA QUIC failed decrypt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.10.158:56414 -> 142.250.180.13
2:443
10/17/2024-14:49:29.741777  [**] [1:2231000:1] SURICATA QUIC failed decrypt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.10.158:56414 -> 142.250.180.13
2:443
10/17/2024-14:49:32.856118  [**] [1:2231000:1] SURICATA QUIC failed decrypt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.10.158:60517 -> 216.58.204.142
:443
10/17/2024-14:49:32.856372  [**] [1:2231000:1] SURICATA QUIC failed decrypt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.10.158:60517 -> 216.58.204.142
:443
10/17/2024-14:49:32.856372  [**] [1:2231000:1] SURICATA QUIC failed decrypt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.10.158:60517 -> 216.58.204.142
:443
10/17/2024-14:49:32.929386  [**] [1:2231000:1] SURICATA QUIC failed decrypt [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.10.158:60517 -> 216.58.204.142
```

```
root@samo:~ # tail -f /var/log/suricata/eve.json
{"timestamp":"2024-10-17T14:49:54.451380+0300","event_type":"stats","stats":{"uptime":515519,"capture":{"kernel_packets":519653,"kernel_drops":0,"kernel_ifdrops":0},"decoder":{"pkts":519653
,"bytes":340830807,"invalid":0,"ipv4":440407,"ipv6":17552,"ethernet":519653,"arp":61668,"unknown_ethertype":26,"chdlc":0,"raw":0,"null":0,"sll":0,"tcp":336230,"udp":116794,"sctp":0,"esp":0,
"icmpv4":442,"icmpv6":1927,"ppp":0,"pppoe":0,"geneve":0,"gre":0,"vlan":0,"vlan_qinq":0,"vlan_qinqinq":0,"vxlan":0,"vntag":0,"ieee8021ah":0,"teredo":0,"ipv4_in_ipv6":0,"ipv6_in_ipv6":0,"mpls
":0,"avg_pkt_size":655,"max_pkt_size":1514,"max_mac_addrs_src":0,"max_mac_addrs_dst":0,"erspan":0,"nsh":0,"event":{"ipv4":{"pkt_too_small":0,"hlen_too_small":0,"iplen_smaller_than_hlen":0,"
trunc_pkt":0,"opt_invalid":0,"opt_invalid_len":0,"opt_malformed":0,"opt_pad_required":2566,"opt_eol_required":0,"opt_duplicate":0,"opt_unknown":0,"wrong_ip_version":0,"icmpv6":0,"frag_pkt_t
oo_large":0,"frag_overlap":0,"frag_ignored":0},"icmpv4":{"pkt_too_small":0,"unknown_type":0,"unknown_code":0,"ipv4_trunc_pkt":0,"ipv4_unknown_ver":0},"icmpv6":{"unknown_type":0,"unknown_cod
e":0,"pkt_too_small":0,"ipv6_unknown_version":0,"ipv6_trunc_pkt":0,"mld_message_with_invalid_hl":0,"unassigned_type":0,"experimentation_type":0},"ipv6":{"pkt_too_small":0,"trunc_pkt":0,"tru
nc_exthdr":0,"exthdr_dupl_fh":0,"exthdr_useless_fh":0,"exthdr_dupl_rh":0,"exthdr_dupl_hh":0,"exthdr_dupl_dh":0,"exthdr_dupl_ah":0,"exthdr_dupl_eh":0,"exthdr_invalid_optlen":0,"wrong_ip_vers
ion":0,"exthdr_ah_res_not_null":0,"hopopts_unknown_opt":0,"hopopts_only_padding":0,"dstopts_unknown_opt":0,"dstopts_only_padding":0,"rh_type_0":0,"zero_len_padn":1742,"fh_non_zero_reserved_
field":0,"data_after_none_header":0,"unknown_next_header":0,"icmpv4":0,"frag_pkt_too_large":0,"frag_overlap":0,"frag_invalid_length":0,"frag_ignored":0,"ipv4_in_ipv6_too_small":0,"ipv4_in_i
pv6_wrong_version":0,"ipv6_in_ipv6_too_small":0,"ipv6_in_ipv6_wrong_version":0},"tcp":{"pkt_too_small":0,"hlen_too_small":0,"invalid_optlen":0,"opt_invalid_len":0,"opt_duplicate":0},"udp":{
"pkt_too_small":0,"hlen_too_small":0,"hlen_invalid":0,"len_invalid":0},"sll":{"pkt_too_small":0},"ethernet":{"pkt_too_small":0},"ppp":{"pkt_too_small":0,"vju_pkt_too_small":0,"ip4_pkt_too_s
mall":0,"ip6_pkt_too_small":0,"wrong_type":0,"unsup_proto":0},"pppoe":{"pkt_too_small":0,"wrong_code":0,"malformed_tags":0},"gre":{"pkt_too_small":0,"wrong_version":0,"version0_recur":0,"ve
rsion0_flags":0,"version0_hdr_too_big":0,"version0_malformed_sre_hdr":0,"version1_chksum":0,"version1_route":0,"version1_ssr":0,"version1_recur":0,"version1_flags":0,"version1_no_key":0,"ve
rsion1_wrong_protocol":0,"version1_malformed_sre_hdr":0,"version1_hdr_too_big":0},"vlan":{"header_too_small":0,"unknown_type":0,"too_many_layers":0},"ieee8021ah":{"header_too_small":0},"vnt
ag":{"header_too_small":0,"unknown_type":0},"ipraw":{"invalid_ip_version":0},"ltnull":{"pkt_too_small":0,"unsupported_type":0},"sctp":{"pkt_too_small":0},"esp":{"pkt_too_small":0},"mpls":{"
header_too_small":0,"pkt_too_small":0,"bad_label_router_alert":0,"bad_label_implicit_null":0,"bad_label_reserved":0,"unknown_payload_type":0},"vxlan":{"unknown_payload_type":0},"geneve":{"u
nknown_payload_type":0},"erspan":{"header_too_small":0,"unsupported_version":0,"too_many_vlan_layers":0},"dce":{"pkt_too_small":0},"chdlc":{"pkt_too_small":0},"nsh":{"header_too_small":0,"u
nsupported_version":0,"bad_header_length":0,"reserved_type":0,"unsupported_type":0,"unknown_payload":0}},"too_many_layers":0},"tcp":{"syn":392,"synack":393,"rst":281,"active_sessions":3,"se
ssions":392,"ssn_memcap_drop":0,"ssn_from_cache":274,"ssn_from_pool":118,"pseudo":24,"pseudo_failed":0,"invalid_checksum":0,"midstream_pickups":0,"pkt_on_wrong_thread":0,"ack_unseen_data":0
,"segment_memcap_drop":0,"segment_from_cache":17708,"segment_from_pool":15178,"stream_depth_reached":25,"reassembly_gap":0,"overlap":3,"overlap_diff_data":0,"insert_data_normal_fail":0,"ins
ert_data_overlap_fail":0,"memuse":1212416,"reassembly_memuse":268288},"flow":{"memcap":0,"total":20271,"active":53,"tcp":588,"udp":19110,"icmpv4":176,"icmpv6":397,"tcp_reuse":0,"get_used":0
,"get_used_eval":0,"get_used_eval_reject":0,"get_used_eval_busy":0,"get_used_failed":0,"wrk":{"spare_sync_avg":99,"spare_sync":198,"spare_sync_incomplete":27,"spare_sync_empty":0,"flows_evi
cted_needs_work":332,"flows_evicted_pkt_inject":370,"flows_evicted":303,"flows_injected":332,"flows_injected_max":0},"end":{"state":{"new":16290,"established":3557,"closed":371,"local_bypas
sed":0},"tcp_state":{"none":0,"syn_sent":0,"syn_recv":0,"established":18,"fin_wait1":0,"fin_wait2":0,"time_wait":0,"last_ack":0,"close_wait":0,"closing":0,"closed":371},"tcp_liberal":0},"mg
r":{"full_hash_pass":8106,"rows_per_sec":6553,"rows_maxlen":2,"flows_checked":49785,"flows_notimeout":29870,"flows_timeout":19915,"flows_evicted":19915,"flows_evicted_needs_work":332},"spar
```