T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SANALLAŞTIRILMIŞ AĞ TOPOLOJİSİ ÜZERİNDE GÜVENLİK DUVARI VE TEHDİT GÖZETLEME SİSTEMLERİNİN OTOMATİZE TEST EDİLMESİ

YÜKSEK LİSANS TEZİ

Bil. Müh. Gökhan ALKAN

Enstitü Anabilim Dalı : BİLG. VE BİL. MÜH.

Tez Danışmanı : Yrd. Doç. Dr. İbrahim ÖZÇELİK

Haziran 2009

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SANALLAŞTIRILMIŞ AĞ TOPOLOJİSİ ÜZERİNDE GÜVENLİK DUVARI VE TEHDİT GÖZETLEME SİSTEMLERİNİN OTOMATİZE TEST EDİLMESİ

YÜKSEK LİSANS TEZİ

Bil.Müh. Gökhan ALKAN

Enstitü Anabilim Dalı : BİLG. VE BİL. MÜH.

Bu tez 17/06/2009 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Prof. Dr. Ümit KOCABIÇAK	Doç. Dr. İsmail ERTÜRK	Yrd. Doç. Dr. İbrahim ÖZÇELİK
Jüri Başkanı	Üye	Üye

TEŞEKKÜR

Bu çalışmanın her aşamasında desteğini hiçbir zaman esirgemeyen hocam Yrd. Doç. Dr. İbrahim ÖZÇELİK'e, yine gerek ders gerekse tez döneminde, benden yardımlarını esirgemeyen Bedirhan URGUN, Muhammed ŞAHİNOĞLU, Cem BAŞKOCAGİL, Nermin ARSLANOĞLU ve Huzeyfe ÖNAL'a teşekkürü bir borç bilirim. Ayrıca sevgili ailemin bütün fertlerine ayrı ayrı sonsuz teşekkürleri iletmemi kendime vazife olarak görüyorum.

İÇİNDEKİLER

KISALTMALAR LİSTESİ

YUM	: Yellowdog Update Manager
DNS	: Domain Name System - İnternet Alan Adı Sistemi
IDS	: Intrusion Detection System - Tehdit Gözetleme Sistemi
GCC	: Gnu Compiler Collection
TCP/IP	: Transmission Control Protocol / Ip Protocol
VPN	: Virtual Private Network
PF	: Packet Filter
RPM	: Redhat Packet Manager
IPS	: Intrusion Prevention System
NAT	: Network Adress Translation - Ağ Adres Çevrimi
NIC	: Network Interface Card - Ağ Arayüz Kartı
GPL	: General Public License
DHCP	: Dynamic Host Configuration Protocol - Dinamik İstemci Yapılandırma Protokolü
SSH	: Secure Shell
NTP	: Network Time Protocol - Ağ Zaman Protokolü
SMTP	: Simple Mail Transmission Protocol - Basit E-Posta Gönderme Protokolü
BIND	: Berkley Internet Name Daemon

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Tcp/Ip, Ağ Güvenliği, Tehdit Gözetleme Sistemi, Güvenlik Duvarı, Python, Scapy

Elektronik dünyasının gelişmesinde rol oynayan en önemli iki unsur; işlemci hızı ve bellek kapasitesidir. Bu iki kıstası belirleyen kaynakların her 18 ayda bir iki katına çıkması bilişim teknolojilerindeki gelişmelere büyük bir ivme kazandırmıştır. Bu gelişmelere paralel olarak, kişisel bilgisayarların her eve girmesi ve son olarak mobil araçların herkes tarafından yaygın olarak kullanılması ile beraber, bilginin dijital ortamdaki yeri daha da önem kazanmıştır. Ayrıca internet dünyasında yaşanan hızlı gelişmeler, bilgiye kolay ve hızlı erişimi kaçınılmaz kılmış, elektrik ve su gibi internet erişimini de insanoğlunun vazgeçilmezleri arasına sokmuştur.

Bankacılık, ticaret, sağlık gibi kişisel ihtiyaçlardan, vergilendirme, nüfus idaresi, eğitim gibi kurumsal ihtiyaçlara kadar birçok işlemin internet üzerinden gerçekleştirilebiliyor olması, bu işlemler sırasında kullanılan bilgilere ve servislere yönelik tehditleri de beraberinde getirmektedir. Kritik uygulamaların üzerinde koştuğu internet alt yapısını oluşturan en temel unsur ağ yapısıdır. Bu nedenle, hassas bilgilere karşı yapılan saldırılar ve bu tehditlere karşı alınması gereken önlemler (ağ güvenliği) büyük oranda ağ bileşenleri ve standartları üzerinde gerçekleştirilmektedir.

Son yıllarda en temel ağ bileşenlerine ve standartlarına yapılan başarılı saldırılar ağ güvenliği konusunda teknik analizlerin ve detaylı akademik çalışmaların gerekliliğini ortaya koymaktadır.

AUTOMATIZION OF INTRUSION DETECTION SYSTEM & FIREWALLS TESTING

SUMMARY

Key Words: Tcp/Ip, Network Security, Intrusion detection system, Firewall, Python, Scapy

Two phenomena playing key roles in the electronical world are; processor speed and memory capacity. Doubling of these resources approximately every 18 months has been giving a substantial acceleration to development of information technologies. In paralel to these improvements, with the ever increasing usage and pervasiveness of personal computers and mobile devices, importance and privacy of the information has become critical in the digital environment. Moreover, the rapid improvements in the Internet resulted in accessing data quickly and more importantly easily, in the end making the Internet Access one of vital resources such as water and electricty.

The fact that nearly all of our needs, from individual ones like banking, commerce, health to institutional ones such as taxes, education, population administration are provided via Internet brings the vast threats against the data and services used by these systems. The single and basic element on which all critical Internet applications run is the network infrastructure. This is the direct consequence of facing malicious attacks and researching against these attacks on network level components and standards.

Extremely critical vulnerabilities found in recent years on the Internet network's most basic standards have prioritized the technical analysis and systematic academical studies on network security.

BÖLÜM 1. GİRİŞ

Internet bağlantı teknolojilerinin (ADSL, WIFI, GPRS) artması ve ucuzlaması ile birlikte bilişim suçlarının sayısı da artmaktadır. Bilişim tehditlerinin önemli bir kısmını içeren ağ güvenliği konusu büyük önem taşımaktadır. Ağ güvenliği alanındaki zafiyetlerin teknik analizlerinin gerçekleştirilmesi, hem mevcut tehditlere alınabilecek olan önlemleri anlamak ve uygulamak için, hem de yeni tehditleri sezmek ve önlemek için altyapı oluşturmakla sağlanacaktır.

Özellikle ağ güvenliğinin öneminin artması ile birlikte, ağ topolojileri içerisinde konuşlandırılan güvenlik duvarı, tehdit gözetleme sistemi gibi uygulamaların da güvenliği de önem kazanmaya başlamıştır.

Elektronik dünyasının gelişmesinde rol oynayan en önemli iki unsur; işlemci hızı ve bellek kapasitesidir. Bu iki kıstası belirleyen kaynakların her 18 ayda bir iki katına çıkması bilişim teknolojilerindeki gelişmelere büyük bir ivme kazandırmıştır. Bu gelişmelere paralel olarak, kişisel bilgisayarların her eve girmesi ve son olarak mobil araçların herkes tarafından yaygın olarak kullanılması ile beraber, bilginin dijital ortamdaki yeri daha da önem kazanmıştır. Ayrıca internet dünyasında yaşanan hızlı gelişmeler, bilgiye kolay ve hızlı erişimi kaçınılmaz kılmış, elektrik ve su gibi internet erişimini de insanoğlunun vazgeçilmezleri arasına sokmuştur.

Bankacılık, ticaret, sağlık gibi kişisel ihtiyaçlardan, vergilendirme, nüfus idaresi, eğitim gibi kurumsal ihtiyaçlara kadar birçok işlemin internet üzerinden gerçekleştirilebiliyor olması, bu işlemler sırasında kullanılan bilgilere ve servislere

yönelik tehditleri de beraberinde getirmektedir. Kritik uygulamaların üzerinde koştuğu internet altyapısını oluşturan en temel unsur ağ yapısıdır. Bu nedenle, hassas bilgilere karşı yapılan saldırılar ve bu tehditlere karşı alınması gereken önlemler (ağ güvenliği) büyük oranda ağ bileşenleri ve standartları üzerinde gerçekleşmektedir.

Son yıllarda en temel ağ bileşenlerine ve standartlarına yapılan başarılı saldırılar bu nedenden dolayı ağ güvenliği konusunda teknik analizlerin ve detaylı akademik çalışmaların gerekliliğini ortaya koymaktadır.

Bu nedenle bu çalışma içerisinde, örnek bir Tcp/Ip ağı üzerinde konuşlandırılmış güvenlik duvarı ve tehdit gözetleme sisteminin test adımlarının otomatize olarak gerçekleştirilmesi işleminin yazılımsal olarak modellenmesi gerçeklenmiştir.

Bu çalışma esnasında tamamen açık kaynak kod yazılımlar tercih edilmiş ve kullanılmıştır. İlgili topolojinin oluşturulmasında, ilgili topoloji içerisinde tercih edilen işletim sistemleri ve bu işletim sistemleri üzerinde koşan uygulamalarda, uygulamanın geliştirilme süreci içerisinde açık kaynak kod yazılımlar tercih edilmiştir. Özellikle açık kaynak kod yazılımların ücretsiz olarak temin edilebilmesi ve uygulama kodlarının incelenebilmesi tercihin birincil sebepleri olmuşlardır.

Uygulamanın gerçeklenmesi için güvelik duvarı ve tehdit gözetleme sistemlerinin konuşlandırıldığı ağ topolojileri incelenmiş, bu topolojiler içerisinden gerçek dünyada da en çok tercih edilen modellerden bir tanesi sanallaştırma teknolojileri yazılımı kullanılarak gerçeklenmiştir.

Bu çalışmanın gerçekleştirilmesi için ihtiyaç duyulan gereksinimler aşağıda belirtilmiştir.

1. Geliştirme ve sunum ortamı
    1.1. Python
    1.2. Scapy
    1.3. Komodo Edit
    1.4. Vmware Workstation
2. İşletim Sistemleri
    2.1. OpenBSD
    2.2. Centos Linux
3. Yardımcı Yazılımlar
    3.1. Tcpdump
    3.2. Pf
    3.3. Snort
    3.4. Wireshark

1.1. Geliştirme ve Sunum Ortamı

Geliştirme ve sunum ortamında sunulan başlıklar ile uygulamanın geliştirilmesinde kullanılan araçlardan bahsedilmiştir. Bunlara ilişkin detaylar aşağıda başlıklar halinde belirtilmiştir.

1.1.1. Python

Python açık kaynak kodlu yarı betik, yarı derlenebilir yazılım platformudur. Web programlama, veritabanı programlama, ağ programlama gibi her türlü amaç için kullanılan Python, özellikle akademik ve Linux/Unix tabanlı uygulamalarda çokça tercih edilmektedir. Hem Windows tabanlı işletim sistemlerinde hem de Linux/Unix tabanlı işletim sistemlerinde uygulama geliştirilebiliyor olması en büyük tercih sebeplerinden birisidir. Python yazılım dili ile nesneye dayalı programlama

gerçekleştirmek mümkündür. Modüler yapısı sayesinde istenilen kütüphaneler ister statik ister dinamik olarak bütünleşik bir biçimde kullanılabilmektedir.

Python programlama diline alternatif olarak Perl ya da Ruby gösterilebilir. Ruby'de Python gibi esnek ve modüler bir betik programlama dilidir. Bunun yanında çok sık kullanılan Perl içinde gerekli modüller kullanılarak yazılım geliştirilebilmektedir. Python yazılım dili, bu çalışma içerisinde, güvenlik duvarı ve tehdit gözetleme sistemlerini otomatize olarak test eden uygulamanın geliştirilmesi için kullanılmıştır.

1.1.2. Scapy

Scapy Tcp/Ip protokol kümesi üzerinde ağ paketleri oluşturmak ve oluşturulan bu paketler üzerinde işlem gerçekleştirebilmek için tasarlanmış Python yazılım dili kütüphanesidir. Sadece kütüphane olarak çalışmasının yanında yalnız başına interaktif olarak kullanılabilmektedir.

Scapy yazılım kütüphanesi, bu çalışma içerisinde Tcp/Ip protokol kümesine ait paket oluşturma, gönderme gibi işlemlerin gerçekleştirilebilmesi için python yazılım dili ile birlikte bütünleşik bir biçimde kullanılmıştır.

1.1.3. Komodo edit

Komodo başta Python olmak üzere Perl gibi birçok yazılım dili için kullanılan yazılım geliştirme ortamıdır. Hem ticari hem de ücretsiz olarak temin etmek mümkündür.

Komodo Edit başta Python olmak üzere birçok programlama dili için geliştirme ortamı sunmaktadır. Alternatif olarak açık kaynak kod dünyasında Emacs gösterilebilir. Bunun yanında ticari yazılım geliştirme ortamlarında kullanılabilmektedir. Komodo Edit, bu çalışma içerisinde uygulama geliştirme ara yüzü olarak kullanılmıştır.

1.1.4. Vmware workstation

İşletim sistemleri ve ağ alt yapılarını sanallaştırma için kullanılan ve bu konuda farklı çözümleri olan bir masaüstü kullanım için öngörülen Vmware uygulamasıdır. Unix ve Linux türevleri başta olmak üzere hemen hemen bütün işletim sistemleri için desteği bulunmaktadır. Bütün bunların yanında özelleştirilmiş ağlarda oluşturmak mümkündür.

Vmware sanallaştırma dünyasında en çok rağbet edilen yazılım olarak göze çarpmaktadır. Hem ücretli hem de ücretsiz kullanım için bulunan yazılımları ile farklı amaçlar için farklı hizmetler sunmaktadır. Özellikle son zamanlarda sanallaştırma ile bakım, devamlılık, yedekleme gibi işlemlerin ucuz ve zahmetsizce gerçekleştirilebilmesinden ötürü, sanallaştırma yazılımlarına olan rağbet gün geçtikçe artmaktadır. Vmware gibi XEN, VirtualBox, QEMU gibi yazılımlar açık kaynak kod dünyasında yerlerini almaktadırlar. Burada Vmware Workstation uygulamasının tercih edilmesindeki en büyük sebep ağ uygulamaları için oldukça esnek şartlar sağlamasıdır.

Vmware Workstation, bu çalışma içerisinde ilgili ağ topolojisinin oluşturulması ve gerçeklenmesi için kullanılmıştır.

1.2. İşletim Sistemleri

İşletim sistemlerinde sunulan başlıklar ile uygulamaların üzerinde koşacağı işletim sistemlerinden bahsedilmiştir. Bunlara ilişkin detaylar aşağıda başlıklar halinde belirtilmiştir.

1.2.1. OpenBSD

4.4BSD tabanlı en güvenilir BSD türevi olarak bilinmektedir. Daha çok güvenlik duvarı (firewall), tehdit gözetleme sistemi (Intrusion Detection System), sanal özel ağlar (Virtual Private Network) gibi güvenlik uygulamalarında tercih edilmektedir. Özellikle pf (Packet Filter) ile bütünleşik biçimde gelmesi, çok kullanılmasının en büyük sebeplerinden bir tanesi olarak göze batmaktadır.

Özellikle güvenlik denince açık kaynak kod dünyasından ilk akla gelen işletim sistemi olan OpenBSD, 10 yılda sadece 2 ciddi açık ile güvenlik konusunda sınıfta kalmıştır. İşletim sistemi olarak oldukça güvenli bir ortam sunmasının yanında pf (Packet Filter) güvenlik duvarı yazılımının esnek kullanımı ile popülaritesi oldukça artmıştır. Alternatif olarak FreeBSD 6.x serisinden sonra pf'in bütünleşik olarak gelmesi ile güvenlik duvarı uygulamalarında kullanımı artmıştır.

OpenBSD işletim sistemi, bu çalışma içerisinde güvenlik duvarı rolündeki sunucu sistem için kullanılmıştır. OpenBSD işletim sistemi ile bütünleşik olarak gelen pf güvenlik duvarı yazılımı olarak kullanılmıştır.

1.2.2. Centos linux

Centos Linux çekirdeğini kullanan bir linux dağıtımıdır. RedHat Enterprise Linux kaynak "*rpm*"'lerinden (Redhat Packet Manager) oluşturulmuş bir dağıtımdır. Alternatif olarak yüzlerce Linux dağıtımı bulunmaktadır ancak RedHat tabanlı olması ve yum paket yönetim sisteminin kullanımının ücretsiz olması nedeniyle Centos dağıtımı tercih edilmiştir.

Centos, Linux bu çalışma içerisinde hem uygulamanın hem de hedef sistemin üzerinde koşacağı işletim sistemi olarak kullanılmıştır.

1.3. Yardımcı Yazılımlar

Yardımcı yazılımlarda sunulan başlıklar ile uygulamanın geliştirilmesi esnasında hata kontrolü ve doğruluk testleri için kullanılan yazılımlardan bahsedilmiştir. Bunlara ilişkin detaylar aşağıda başlıklar halinde belirtilmiştir.

1.3.1. Tcpdump

Hemen hemen bütün Unix ve linux türevleri ile birlikte ön tanımlı olarak gelen ve komut satırından çalışan, detaylı paket analizi gerçekleştirmek için kullanılan açık kaynak kodlu bir yazılımdır. Aşağıda basit bir kullanımı ve örnek çıktısı görülmektedir.

```
# tcpdump -tttnn -i em0 -s 1024 host 192.168.1.2 and port 22
000000 IP 192.168.1.1.22 > 192.168.1.2.1116: P
2548453184:2548453380(196) ack
1937927393 win 33370 <nop,nop,timestamp 2058517610 15146>
023938 IP 192.168.1.2.1116 > 192.168.1.1.22: . ack 196 win 64715
<nop,nop,timestamp
15146 2058517606>
#
```

Hemen hemen bütün Linux/Unix dağıtımları ile birlikte gelmesi nedeniyle alternatifsiz ağ paket analiz yazılımı olarak kullanılmaktadır.

Tcpdump, bu çalışma içerisinde Tcp/Ip protokol kümesine ait oluşturulup gönderilen ya da alınan paketlerin hedef sistemlerdeki görüntüsünün alınması için kullanılmıştır.

1.3.2. Pf

OpenBSD işletim sistemi ile ön tanımlı olarak gelen pf, FreeBSD işletim sistemi içinde kullanılabilir hale gelmiştir. Aşağıda örnek bir kural tanımı ve açıklaması yer almaktadır.

```
block drop in on em0 proto tcp from 192.168.1.1
// em0 ara yüzüne 192.168.1.1 ip adresinden gelen bütün tcp paketlerini
engelle.
```

Pf, bu çalışma içerisinde güvenlik duvarı rolündeki sunucu sistem üzerinde koşan, güvenlik duvarı uygulaması olarak kullanılmıştır.

1.3.3. Snort

Tehdit gözetleme sistemi özelliklerinin yanında çeşitli yazılımlar ile bütünleşik olarak tehdit engelleme sistemi (Intrusion Prevention System) olarak da görev yapabilen açık kaynak kodlu bir yazılımdır. Snort, bu çalışma içerisinde tehdit gözetleme sistemi yazılımı olarak kullanılmıştır.

1.3.4. Wireshark

Ethereal yazılımının 0.99.0 sürümünden sonra Wireshark adı ile devam eden detaylı paket analizi gerçekleştirmek için kullanılan Linux/Unix ve Windows ortamlarında kullanılabilen açık kaynak kodlu bir yazılımdır.

Wireshark, bu çalışma içerisinde Tcp/Ip protokol kümesine ait oluşturulup gönderilen ya da alınan paketlerin hedef sistemlerdeki görüntüsünün alınması için kullanılmıştır.

1.4. Tez Organizasyonu

Tezin birinci bölümünde; tez çalışmasının amacı ve bu çalışma da kullanılacak olan gereksinimlerden bahsedilmiştir.

Tezin ikinci bölümünde; güvenlik duvarı ve tehdit gözetleme sisteminin test adımlarının otomatize olarak gerçekleştirilmesi için gerekli sanal laboratuar ortamının oluşturulması, ağ topolojisi ve bu ağ topolojisi üzerinde yer alan sunucu sistemlerin kurulumu, yapılandırılması ve yine bu sistemler üzerinde koşacak olan uygulamaların kurulum ve yapılandırma adımlarından bahsedilmiştir.

Tezin üçüncü bölümünde; uygulamanın gerçekleştirileceği yazılım kütüphanesinin kurulumu ve detaylı kullanımından bahsedilmiştir.

Tezin dördüncü bölümünde; tez uygulaması çalıştırılarak, uygulamaya ait detaylardan bahsedilmiş bunun yanında güvenlik duvarı ve tehdit gözetleme sisteminin test adımları örnek senaryolar ile birlikte otomatik olarak gerçekleştirilmiştir.

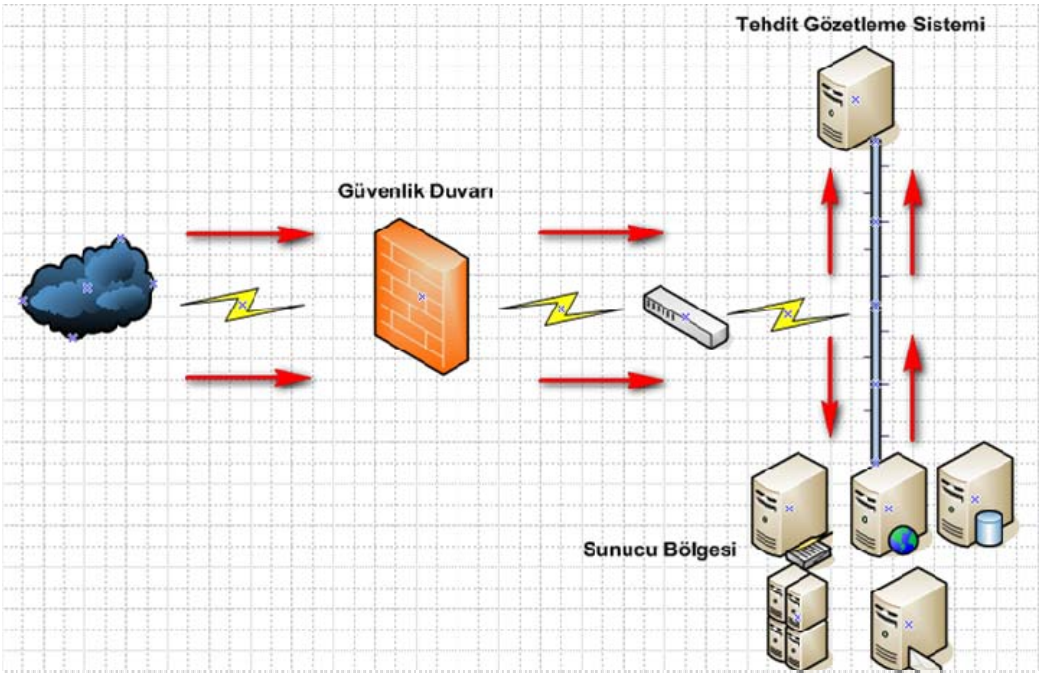
BÖLÜM 2. AĞ TOPOLOJİSİNİN OLUŞTURULMASI

Çok hızlı bir biçimde gelişen internet dünyasın da gün geçtikçe iyi niyetli olarak tasarlanan protokol ya da uygulamalar kullanım alanının artması ile birlikte kötü niyetli kişiler tarafından kötüye kullanılmaya başlanıyor. Bu hırsız polis kovalamacası içerisinde kötü niyetli kişiler kadar güvenlik uzmanları da savunma amaçlı çalışmaları gerçekleştiriyorlar. Bütün bunların ışığın da ağ güvenliğinin ne kadar önemli olduğu gerçeği bir kere vurgulanmaktadır.

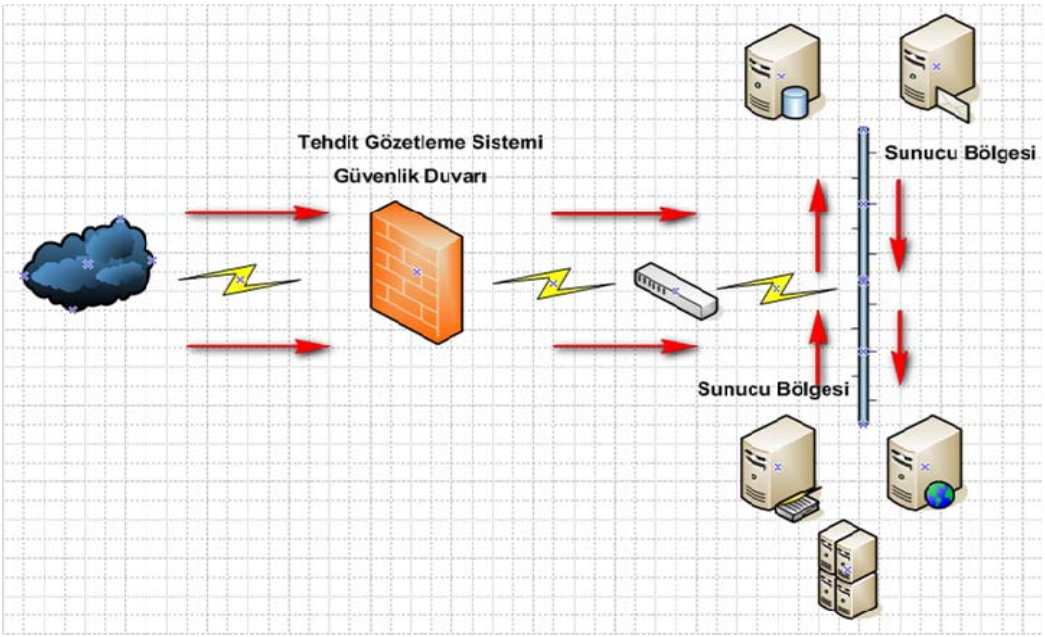
Özellikle günümüz topolojilerin de olmaz ise olmazlardan biri olarak yerlerini alan güvenlik duvarları ve tehdit gözetleme sistemleri, gerçek zamanlı olarak topoloji içerisinde yerlerini almaktadırlar. Özellikle güvenlik duvarı uygulamaları, gerek yazılım tabanlı gerek ise donanım tabanlı, kurumsal topolojilerden küçük ölçekli topolojilere kadar kullanımı zorunlu hale gelmiştir.

Güvenlik duvarları genel olarak konuşlandırıldığı yere göre gelen ve giden ağ trafiği üzerinde yapılandırmasında belirlenen kurallara uygun olarak denetim sağlamak için kullanılırlar. Güvenlik duvarlarının durum korumalı ve durum korumasız olmak üzere 2 temel türü bulunmaktadır. Tehdit gözetleme sistemleri de güvenlik duvarı gibi yapılandırmasında belirlenen kurallara uygun olarak konuşlandırıldığı yere göre gelen ve giden ağ trafiği üzerindeki tehditleri sezinlemek için kullanılırlar.

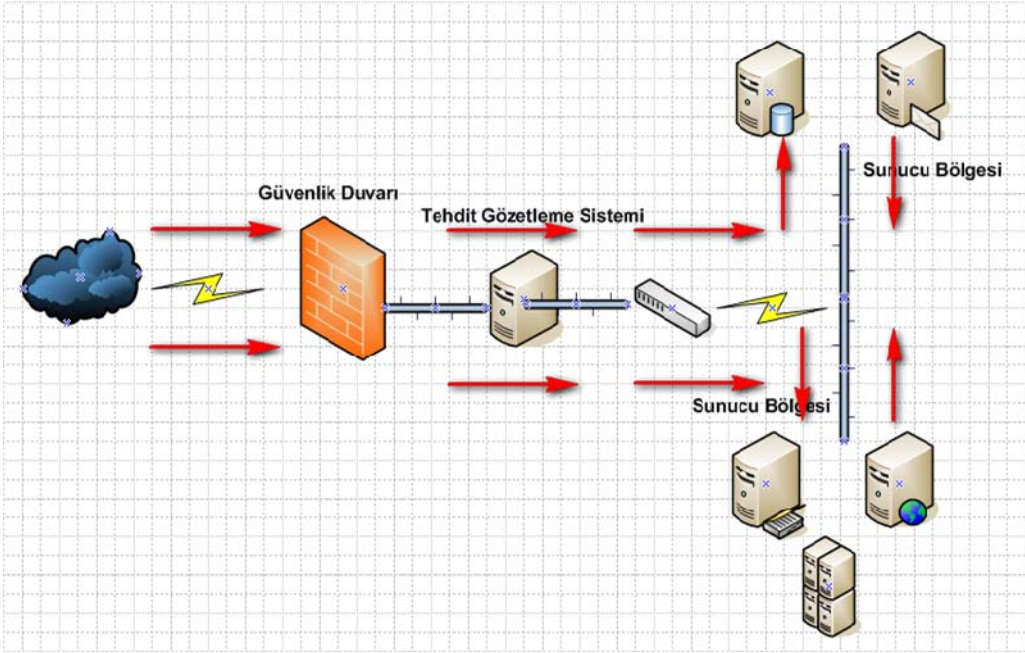
Güvenlik duvarı ve tehdit gözetleme sistemlerinin topolojiler içerisin de ağ trafiğini kontrol etmek için çeşitli konumları bulunmaktadır. Güvenlik duvarı ve tehdit gözetleme sistemlerinin konuşlandırılmasında en çok tercih edilen topolojiler Şekil 2.1, Şekil 2.2 ve Şekil 2.3’ de gösterilmiştir.



Şekil 2.1. Güvenlik Duvarı ve Tehdit Gözetleme Sistemlerinin Ağ Topolojileri İçerisindeki Muhtemel Konumları - 1

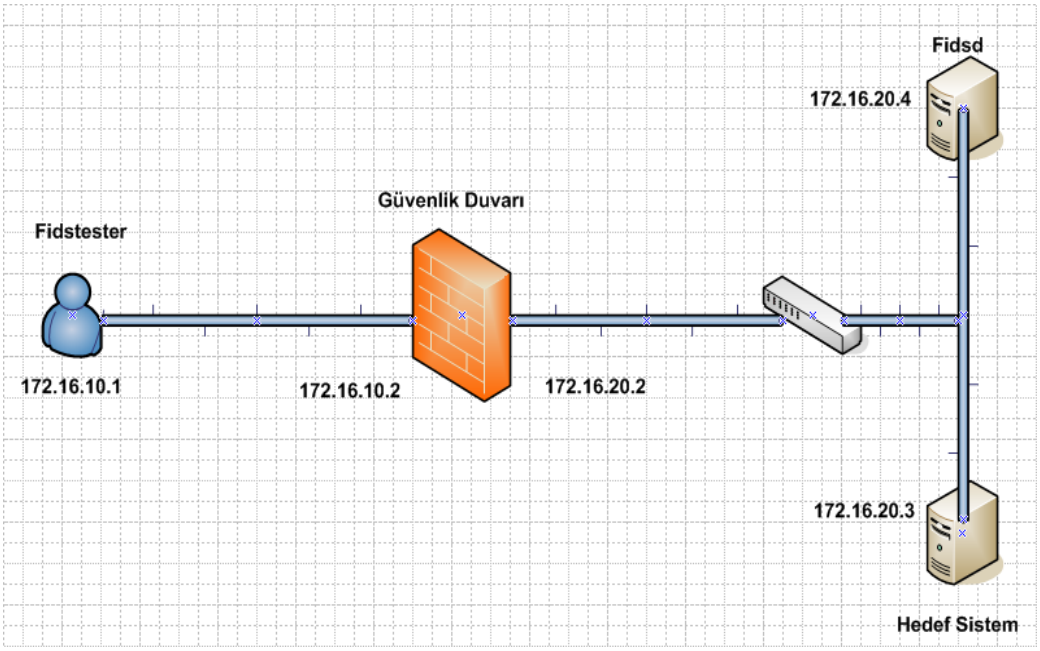


Şekil 2.2. Güvenlik Duvarı ve Tehdit Gözetleme Sistemlerinin Ağ Topolojileri İçerisindeki Muhtemel Konumları - 2



Şekil 2.3. Güvenlik Duvarı ve Tehdit Gözetleme Sistemlerinin Ağ Topolojileri İçerisindeki Muhtemel Konumları - 3

Bu çalışma da Şekil 2.1, Şekil 2.2 ve Şekil 2.3'de gösterilen topolojiler içerisinden en çok tercih edilen topolojilerden bir tanesi olan Şekil 2.1'de gösterilen topoloji kullanılmıştır. Çalışma içerisin de kullanılacak olan topoloji içerisinde yer alan sistemlere ait ip adres bilgisi ve rolleri Şekil 2.4'de gösterilen topolojideki gibi olmaktadır.



Şekil 2.4 Genel Ağ Topolojisi

Ağ topolojisinin gerçeklenmesi için birden fazla cihaza ihtiyaç duyulması gerekmektedir. Bu hem maliyet hem de taşınabilirlik olmak üzere bir takım problemleri de beraberinde getirmektedir. Bu işlemin tek bir bilgisayar üzerinde gerçeklenmesi için sanallaştırma yazılımı olan Vmware Workstation kullanılmış ve topolojinin gerçeklenmesin de kullanılmıştır. Burada belirtilen topoloji güvenlik duvarı ve tehdit gözetleme sistemlerinin konuşlandırılmasın da tercih edilen genel bir topolojiyi içermektedir.

2.1. Wmware Workstation Kurulumu ve Performans Yapılandırması

Vmware sanallaştırma dünyasında en çok rağbet edilen yazılım olarak göze çarpmaktadır. Hem ücretli hem de ücretsiz kullanım için bulunan yazılımları ile farklı amaçlar için farklı hizmetler sunmaktadır. Özellikle son zamanlarda sanallaştırma ile bakım, devamlılık, yedekleme gibi işlemlerin ucuz ve zahmetsizce gerçekleştirilebilmesinden ötürü, sanallaştırma yazılımlarına olan rağbet gün geçtikçe artmaktadır. Vmware gibi XEN, VirtualBox, QEMU gibi yazılımlar açık

kaynak kod dünyasında yerlerini almaktadırlar. Burada Vmware Workstation uygulamasının tercih edilmesindeki en büyük sebep ağ uygulamaları için oldukça esnek şartlar sağlamasıdır.

Vmware Workstation işletim sistemleri ve ağ alt yapılarını sanallaştırma için kullanılan ve bu konuda farklı çözümleri olan bir masaüstü kullanım için öngörülen Vmware uygulamasıdır. Unix ve Linux türevleri başta olmak üzere hemen hemen bütün işletim sistemleri için desteği bulunmaktadır. Bütün bunların yanında özelleştirilmiş ağlarda oluşturmak mümkündür.

Ağ topolojisinin oluşturulmasında kullanılan gerekli donanım özellikleri ve yazılımların sürüm numaraları Tablo 2.1 ve Tablo 2.2'de gösterildiği gibi olmaktadır.

Tablo 2.1. Donanım Özellikleri

İşlemci	Intel(R) Core(TM) Duo CPU 1.86GHz
Bellek	4 GB (En az 2Gb)

Tablo 2.2. Kullanılan Yazılımlar ve Sürüm Numaraları

Sunucu İşletim Sistemi	Fedora 9
Sunucu İşletim Sistemi Masaüstü Uygulaması	Xfce
Misafir İşletim Sistemi	OpenBSD4.4 (Pf), Centos 5.2
Vmware Workstation Sürüm Numarası	6.5.1–126130
Çekirdek Sürüm Numarası	2.6.27.12–78.2.8

Sunucu işletim sisteminin "*cd*" medya ortamından kurulumu esnasında KDE (K Desktop Environment) ve GNOME (GNU Network Object Model Environment)

olmak üzere 2 adet masaüstü yönetici uygulaması sunulmaktadır. Gerçeklenecek olan ağ topolojisi için 3 adet misafir işletim sistemi kurulacağından ötürü, harcanacak hafıza miktarını en aza indirmek için sunucu işletim sistemi üzerinde masaüstü yöneticisi olarak Xfce tercih edilmiştir. Kurulum detayları ve nasıl etkin hale getirileceği aşağıda belirtilmiştir.

```
# yum groupinstall xfce4

# yum install switchdesk-gui
$ switchdesk-helper XFCE
```

2.1.1. Vmware Workstation kurulumu

Vmware Workstation yazılımı kurulmadan önce bazı paketlerin sisteme kurulması gerekmektedir. Bu paketlerin sisteme kurulumu için yum (Yellowdog Updater) paket yönetim sistemi kullanılmıştır. Bu paketlerin kurulumu için aşağıdaki adım izlenmelidir.

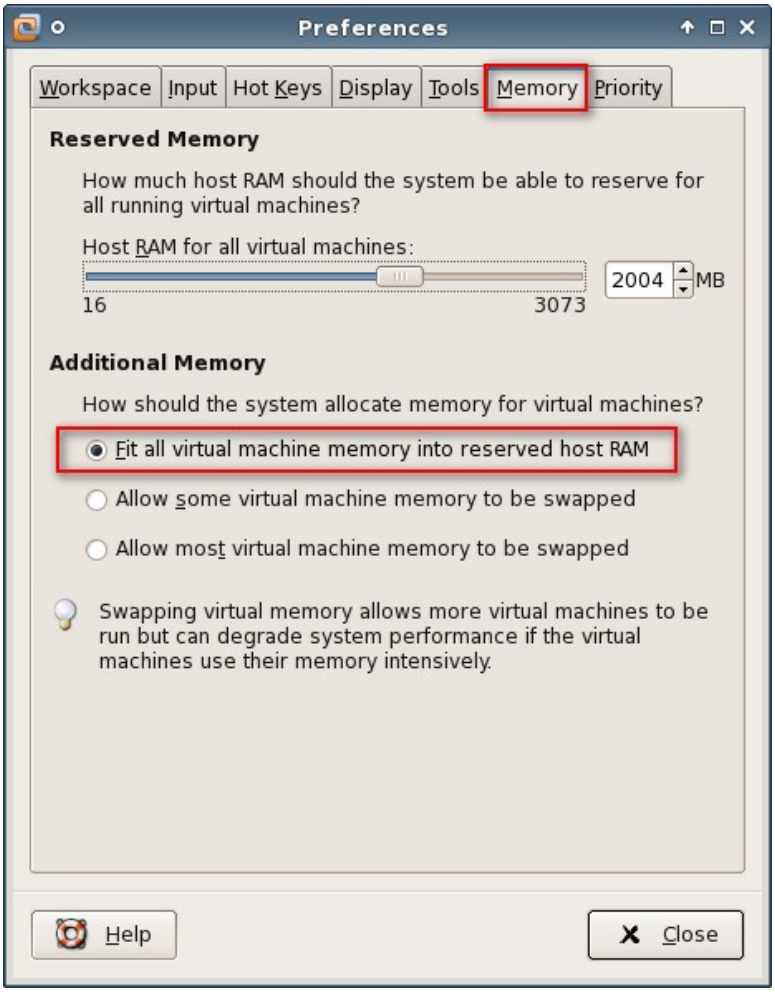
```
# yum install kernel-devel kernel-header make gcc gcc-c++
```

Vmware Workstation yazılımını sisteme kurmak için "*rpm*" komutu ilgili parametreler ile çalıştırılarak kurulum gerçekleştirilir.

```
# rpm -ivh Vmware-Workstation-6.5.1-126130.i386.rpm
Preparing...                ########################################### [100%]
1:VmwareWorkstation
########################################### [100%]
 #
```

Vmware Workstation uygulamasında misafir işletim sistemleri ön tanımlı olarak sunucu işletim sisteminin sahip olduğu bütün bellek miktarını harcamaktadır. Burada kullanılacak ağ topolojisi üzerinde 3 adet misafir işletim sistemi kullanılacağı düşünüldüğünde, bellek miktarının kullanımında artış olabileceği ve sunucu işletim sistemi üzerinde uygulama geliştirme sırasında problemler yaşama olanağı artacağı için, misafir işletim sistemleri tarafından kullanılacak olan bellek miktarı kullanımı sınırlandırılmalıdır. Bunun için aşağıdaki adımlar sırası ile takip edilerek uygulanmalıdır.

"*Edit*" → "*Preferences*" → "*Memory*" seçeneklerinden "*Host BELLEK for all virtual machines*" bölümünden ne kadar bellek miktarı kullanılması gerektiği seçilmelidir. Duruma ilişkin ilgili bir örnek Şekil 2.5' de gösterilmiştir.



Şekil 2.5. Misafir İşletim Sistemleri Bellek Miktarı Sınırlandırılması

Bu işlemin gerçekleştirebilmesi için Vmware Workstation uygulaması "*root*" kullanıcı hakları ile çalıştırılmalıdır.

2.2. Vmware ile İlgili Tanımlar ve Ağ Bağlantı Seçenekleri

Misafir işletim sistemi, Vmware Workstation içerisine kurulan işletim sistemlerini içeren tanımlamadır. Sunucu işletim sistemi, Vmware Workstation yazılımının kurulduğu işletim sistemini içeren tanımlamadır. Burada sunucu işletim sistemi olarak Fedora 9 kullanılmıştır.

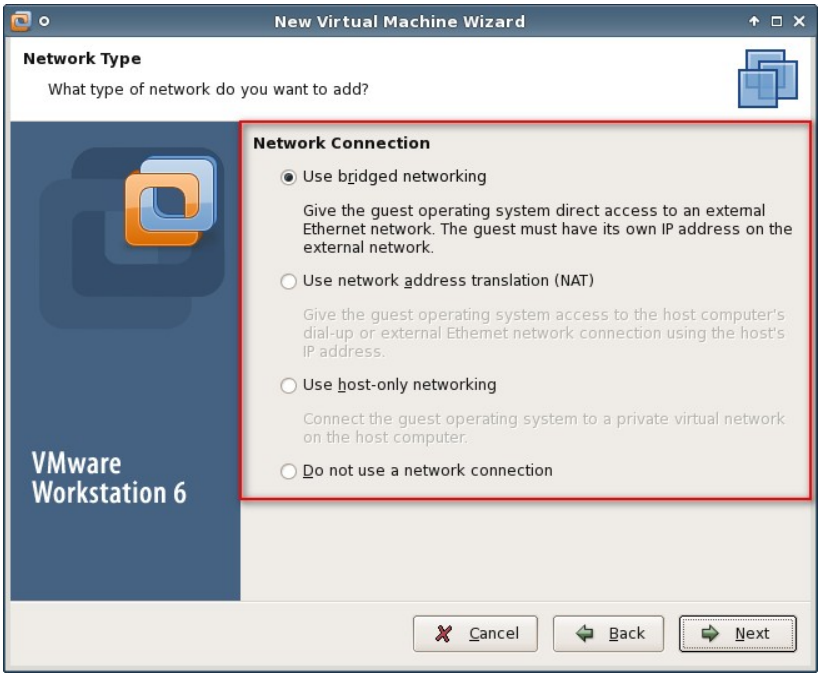
2.2.1. Vmware ağ bağlantı seçenekleri

Misafir işletim sistemi için kullanılacak olan ağ bağlantı seçeneğine verilen tanımlamadır. Bu şekilde misafir işletim sisteminin diğer misafir işletim sistemleri ve sunucu işletim sistemi ile nasıl haberleşeceği belirlenmektedir.

2.2.2. Vmware ağ bağlantı yapılandırması

Misafir işletim sisteminin kullanacağı ağ bağlantı seçenekleri için birkaç yöntem bulunmaktadır. "*Bridge network*", "*host-only*" ve "*nat*" (Network Adress Translation) olmak üzere 3 adet ağ bağlantı seçeneği bulunmaktadır. Bunların dışında istenirse ağ bağlantı seçeneği kullanılmayarak ağ erişimi gerçekleştirmeden de misafir işletim sistemi kurularak kullanılmaktadır.

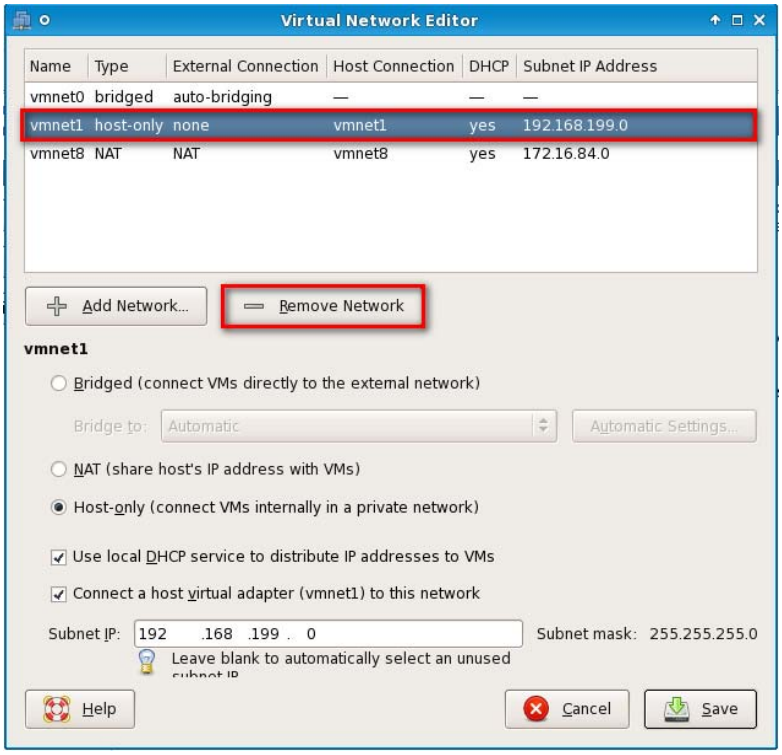
Host only bağlantı türünde misafir işletim sistemi sadece sunucu işletim sistemi ile özel bir ip adres bloğu üzerinden iletişim kurmaktadır. Vmware Workstation için kullanılabilen ağ bağlantı seçenekleri Şekil 2.6' de gösterilmektedir.



Şekil 2.6. Vmware Workstation Ağ Bağlantı Seçenek Ara Yüzü

2.2.3. Topoloji üzerindeki misafir işletim sistemleri ağ yapılandırması

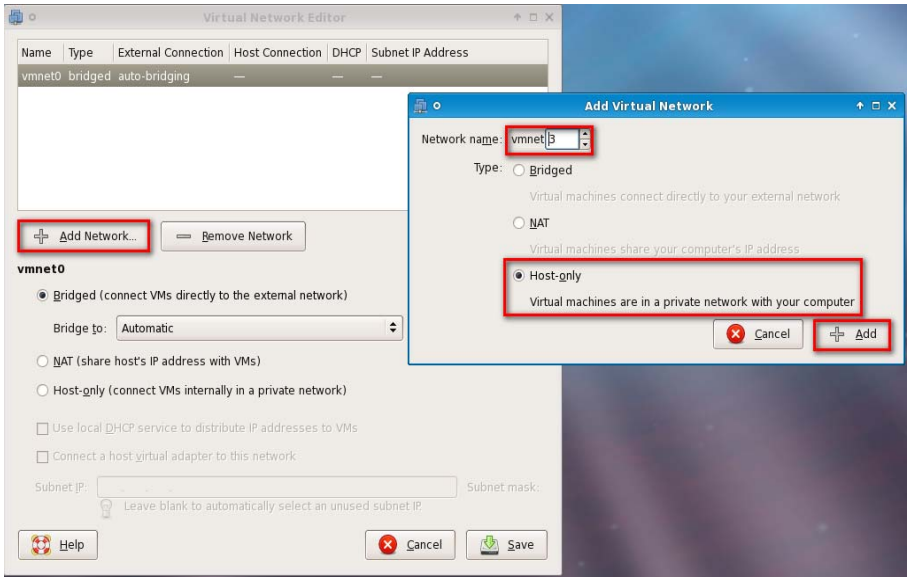
Vmware Workstation uygulamasının kurulumunun ardından misafir işletim sistemleri için gerekli ağ yapılandırması için gerekli sanal aygıtlar oluşturulmalı ve Tcp/Ip yapılandırmaları gerçekleştirilmelidir. Tehdit gözetleme sistemi olarak görev yapacak misafir işletim sisteminin, kurban rolündeki misafir sisteme gelen ve giden tüm ağ trafiğini dinleyebilmesi için, Vmware Workstation uygulaması üzerinde ön tanımlı olarak gelen sanal aygıtlar kaldırılmalı, bunun yerine yeni sanal aygıtlar oluşturulmalı ve uygun bir biçimde yapılandırılmalıdır. Bunun için "*Edit*" → "*Virtual Network*" yolundan Vmware ağ yapılandırmasını yönetmek için gerekli panel ekrana gelecektir. Buradan ön tanımlı olarak gelen "*vmnet1*" ve "*vmnet8*" sanal ağ aygıtları kaldırılmalıdır. Bu işlemlerin nasıl gerçekleştirileceği Şekil 2.7, Şekil 2.8, Şekil 2.9, Şekil 2.10, Şekil 2.11 ve Şekil 2.12'da sırası ile gösterilmektedir.



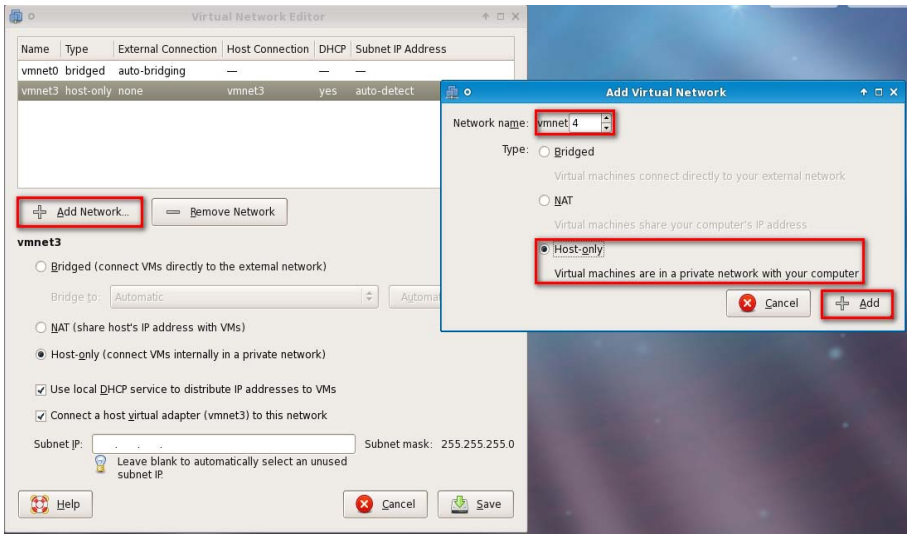
Şekil 2.7. Misafir İşletim Sistemi Ağ Yapılandırması - 1

Şekil 2.8. Misafir İşletim Sistemi Ağ Yapılandırması - 2

Ön tanımlı Vmware sanal ağ aygıtlarının kaldırılmasının ardından ağ topolojisi üzerinde kullanılacak olan misafir işletim sistemleri için gerekli ağ aygıtları oluşturulmalı ve gerekli yapılandırmalar gerçekleştirilmelidir. Bunun için 2 adet "*host only*" sanal Vmware Workstation ağ aygıtı oluşturulmalıdır. Bu işlemleri gerçekleştirmek için aşağıdaki adımlar sırası ile takip edilmelidir.

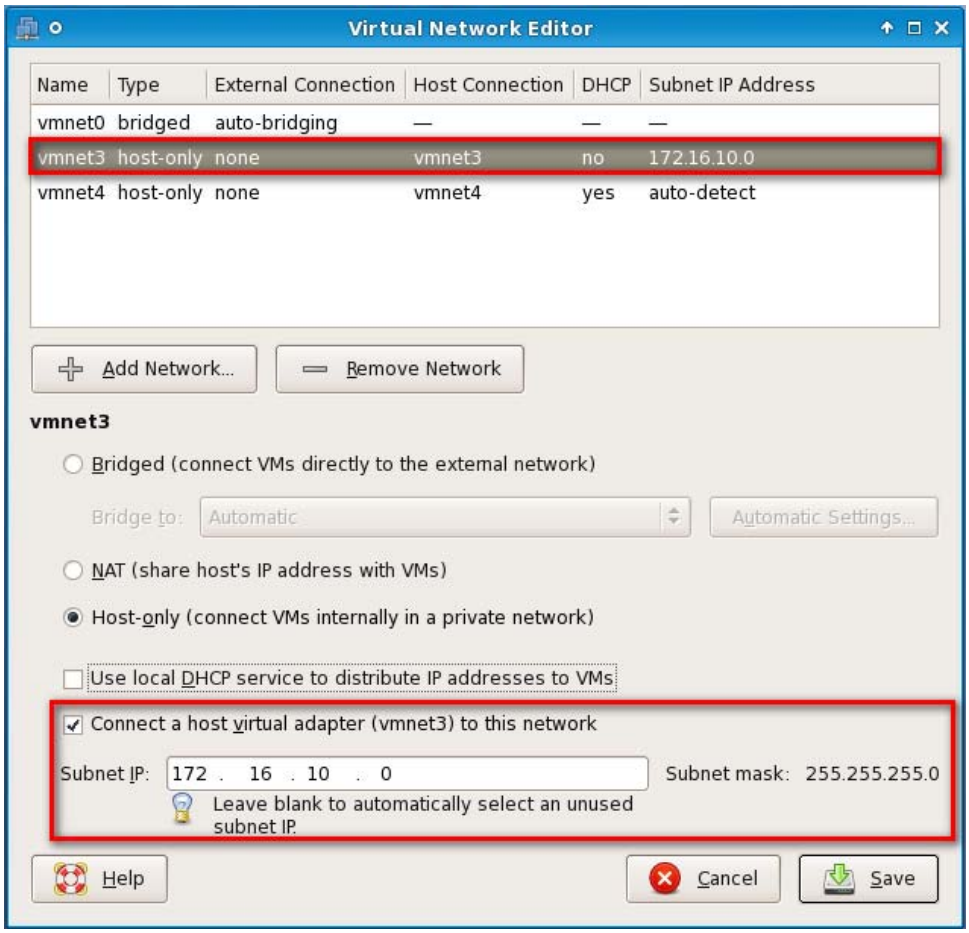


Şekil 2.9. Misafir İşletim Sistemi Ağ Yapılandırması - 3

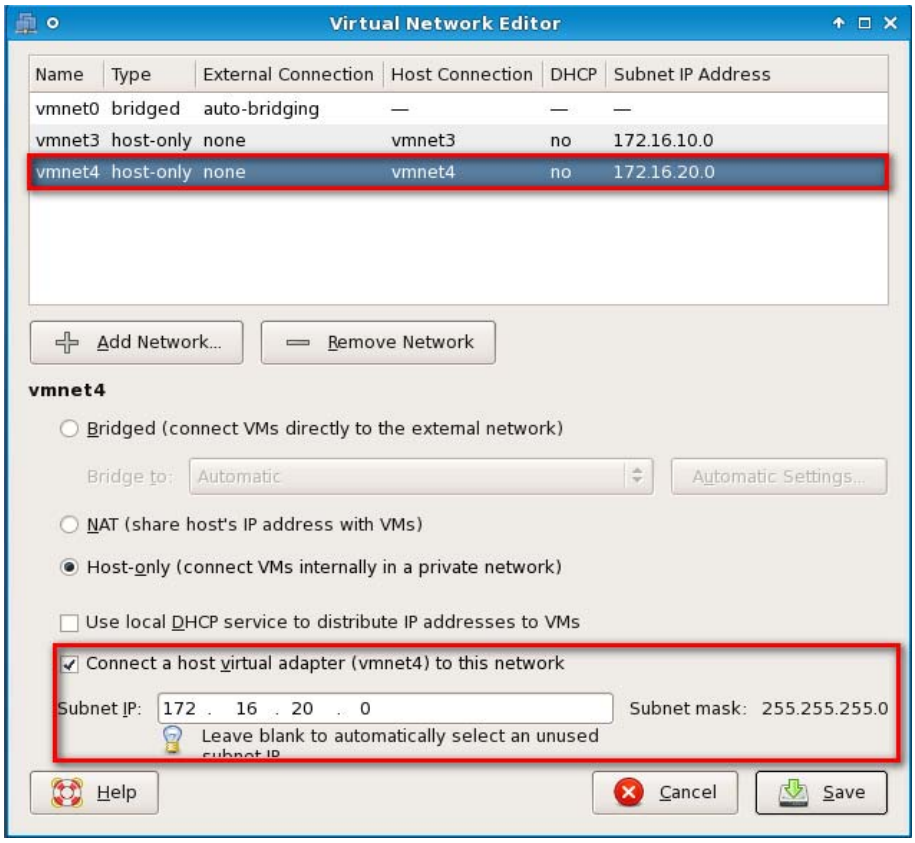


Şekil 2.10. Misafir İşletim Sistemi Ağ Yapılandırması - 4

Misafir işletim sistemleri için gerekli sanal ağ aygıtlarının oluşturulmasının ardından bu aygıtlar için gerekli ağ yapılandırması gerçekleştirilmelidir. Bu işlem yine "*virtual network*" paneli aracılığı ile gerçekleştirilebilmektedir.



Şekil 2.11. Misafir İşletim Sistemi Ağ Yapılandırması - 5



Şekil 2.12. Misafir İşletim Sistemi Ağ Yapılandırması - 6

Gerekli Vmware sanal ağ aygıtlarının oluşturulmasının ardından ilgili aygıtlar sunucu sistem üzerinde "*ifconfig*" komutu yardımı ile görüntülenebilmektedir. Bu durum Şekil 2.13'de görülmektedir.

```
vmnet3    Link encap:Ethernet  HWaddr 00:50:56:C0:00:03
          inet addr:172.16.10.1  Bcast:172.16.10.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fec0:3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

vmnet4    Link encap:Ethernet  HWaddr 00:50:56:C0:00:04
          inet addr:172.16.20.1  Bcast:172.16.20.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fec0:4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Şekil 2.13. Oluşturulan Vmware Workstation Sanal Ağ Aygıtlarının Sunucu Sistem Üzerinde Görüntülenmesi

İlgili ağ topolojisi üzerinde misafir işletim sistemi için kullanılacak ağ ara yüzleri, atanmış ip adres bloğu ve sunucu sistemlerin görevleri Tablo 2.3'de belirtildiği gibi olmaktadır.

Tablo 2.3. Kullanılan İşletim Sistemleri, Rolleri ve Belirtimleri

İşletim Sistemi	**Rolü**	**Vmware Ağ Ara yüzü – Ağ Ara yüzü**	**İp Adres Bilgisi**
Sunucu (Fedora 9)	Fidstester - Saldırgan	eth0 vmnet3 vmnet4	- 172.16.10.1/24 172.16.20.1/24
Misafir (OpenBSD)	Güvenlik Duvarı	vic0 (vmnet3) vic1 (vmmnet4)	172.16.10.2/24 172.16.20.2/24
Misafir (Centos)	Kurban	eth0 (vmnet4)	172.16.20.4/24
Misafir (Centos)	Tehdit Gözetleme Sistemi	eth0 (vmnet4)	172.16.20.3/24

2.3. Vmware Yapılandırması ve Sistem Açılışın da Aktif hale Getirilmesi

Vmware Workstation uygulamasının bilgisayara kurulumunun ardından sistemi başlatmak ve aktif hale getirmek için aşağıdaki işlemlerin sırası ile uygulanması gerekmektedir.

```
# /etc/init.d/Vmware start
…
# chkconfig Vmware on
```

2.3.1. Vmware ağ yapılandırmasının doğrulanması ve sunucu sistem üzerinde uygulanması gereken adımlar

Gerekli ağ aygıtlarının oluşturulmasının ardından sunucu sistem üzerinde "*route*" komutu ilgili parametre ile çalıştırılarak yönlendirme tablosu görüntülenebilir.

```
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.20.0     0.0.0.0         255.255.255.0   U     0      0        0 vmnet4
172.16.10.0     0.0.0.0         255.255.255.0   U     0      0        0 vmnet3
```

Görüldüğü gibi 172.16.20.0/24 ip adres bloğu için "*vmnet4*" ara yüzü kullanılmaktadır. Bu durumda saldırgan sistemden bu ip adres bloğuna giden bütün paketler "*vmnet4*" ara yüzü üzerinden direkt olarak gidecektir. Hâlbuki ağ topolojisi üzerinde bu paketlerin "*vmnet3*" ara yüzü üzerinden gitmesi istenmektedir. Bunun için bu yönlendirme silinip yerine bu ağ için gidecek olan paketlerin 172.16.10.2 ara yüzüne gitmesini sağlayacak yönlendirme belirtilmelidir.

```
# route del -net 172.16.20.0/24
# route add -net 172.16.20.0/24 gw 172.16.10.2
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.20.0     172.16.10.2     255.255.255.0   UG    0      0        0 vmnet3
172.16.10.0     0.0.0.0         255.255.255.0   U     0      0        0 vmnet3
```

Görüldüğü gibi artık 172.16.20.0/24 ağına giden bütün paketler 172.16.10.2, yani güvenlik duvarı rolündeki misafir işletim sisteminin saldırgan ile iletişime geçeceği ağ ara yüzüne iletilecektir.

Eğer Vmware Workstation uygulaması "*root*" harici bir kullanıcı ile çalıştırılıyor ise kurulan misafir işletim sistemleri "*promiscuous*" moda alınmak istendiğinde, daha önce oluşturulan "*custom: spesific virtual network*" ethernet ara yüzüne Vmware Workstation uygulamasını çalıştıran kullanıcı tarafından okuma ve yazma haklarının bulunması gerekmektedir. Ön tanımlı olarak bu özel aygıt sadece "*root*" kullanıcısı tarafından okunabilir ve yazılabilir durumdadır.

Vmwware Workstation uygulamasının galkan kullanıcısının çalıştırdığı varsayılırsa eğer bu durumu çözmek için aşağıdaki adımların sırası ile uygulanması gerekmektedir.

```
# ls -la /dev/vmnet2
crw------- 1 root root 119, 2 2008-11-15 13:58 /dev/vmnet2
# chown galkan:root /dev/vmnet2
# chmod 660 /dev/vmnet2
```

2.4. Misafir İşletim Sistemleri Kurulumu

Ağ topolojisinin oluşturulmasında kullanılacak olan aynı zamanda Vmware Workstation uygulaması içerisinde kurulum adımları gerçekleştirilecek olan misafir işletim sistemleridir. Misafir işletim sistemlerinin kurulum adımları ve detayları ilgili başlıklar altında detaylı olarak gösterilmiştir.

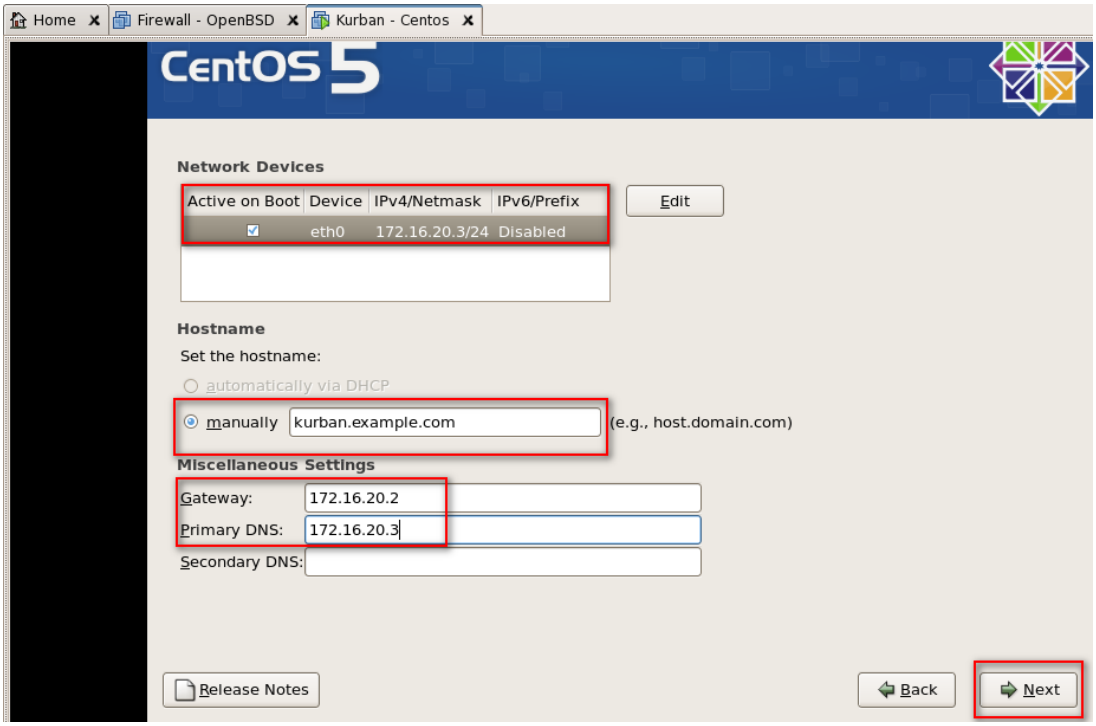
2.4.1. Tehdit gözetleme ve kurban rolündeki misafir işletim sistemlerinin kurulumu

Şekil 2.4'de bulunan topolojide belirtildiği üzere kurban ve tehdit gözetleme rolündeki sunucu sistemleri, Centos 5.2 işletim sistemine ve tek ağ ara yüzüne sahip olacaklardır. Bu ara yüz "*host-only*" olarak çalışacaktır. Misafir işletim sistemi ile ilgili ayrıntılar Tablo 2.4'de gösterilmektedir.

Tablo 2.4. Misafir İşletim Sistemi Özellikleri

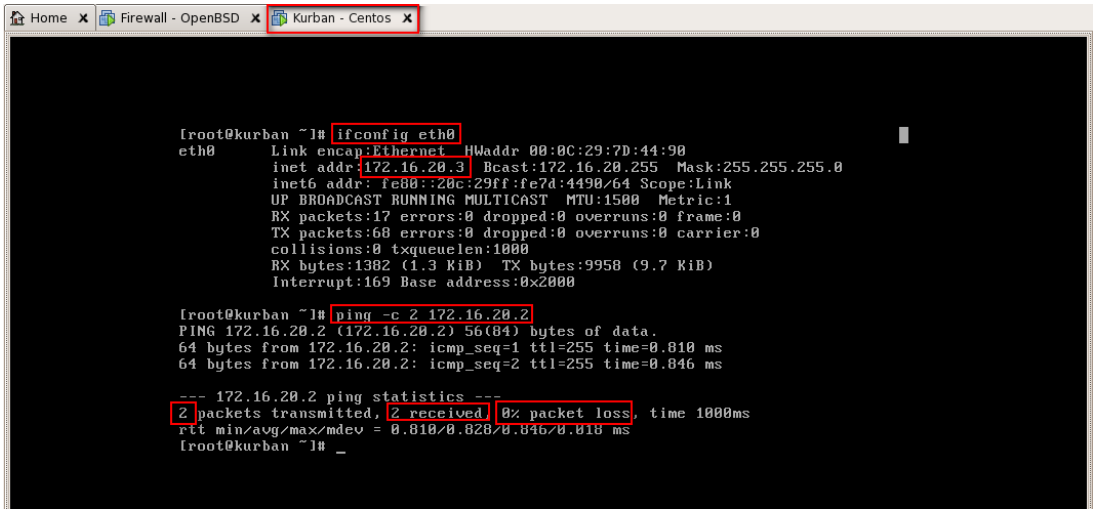
Rolü	Tehdit Gözetleme Sistemi - Kurban
İşletim Sistemi	Centos5.2
Ağ Ara yüzleri	vmnet4 (custom spesific – hub)
İp Adres/Adres Bloğu	Kurban vmnet4 -> 172.16.20.3 Tehdit Gözetleme Sistemi vmnet4 -> 172.16.20.4

Vmware Workstation uygulaması içerisinde gerçekleştirilmesi gereken adımların tamamlanmasının ardından, Centos Linux işletim sistemi kurulum adımları başlatılır ve uygun adımları gerçekleştirilerek kurulum başlatılır. Kurulumun esnasında Tcp/Ip yapılandırmasına ilişkin ayarların gerçekleştirildiği ekran görülmektedir. İp adresi bilgisi için 172.16.20.3, ön tanımlı ağ geçidi için 172.16.20.2, dns (Domain Name System) sunucu ip adresi bilgisi için 172.16.20.3 ve son olarak makine ismi olarak da "*kurban.example.com*" kullanılmaktadır. Bu durum Şekil 2.14'de gösterilmektedir.



Şekil 2.14. Kurban Rolündeki Misafir İşletim Sistemi Tcp/Ip Yapılandırması

Son olarak "*reboot*" komutu ile sistem yeniden başlatılarak gerçekleştirilen değişikliklerin etkin hale gelmesi sağlanmalıdır. Kurulum işleminin tamamlanmasının ardından Tcp/Ip yapılandırması ile ilgili belirtimler Şekil 2.15'de gösterildiği gibi kontrol edilmelidir.



Şekil 2.15. Kurban Rolündeki Misafir İşletim Sistemi Tcp/Ip Yapılandırması Doğrulanması

Tehdit gözetleme sistemi rolündeki misafir işletim sistemi için gerçekleştirilmesi gereken kurulum adımları kurban rolündeki misafir işletim sistemi kurulum adımları ile aynı olmaktadır. Tcp/Ip yapılandırmasında farklılık gösterecek tek ayar ip adres bilgisinin 172.16.20.4 olacak şekilde olmasıdır.

2.4.2. Güvenlik duvarı rolündeki misafir işletim sistemi kurulumu

Ağ topolojisinde belirtildiği üzere güvenlik duvarı rolü, OpenBSD4.3 işletim sistemi ile sağlanacak ve 2 ağ ara yüzüne sahip olacak bir sistemdir. Bu ağ ara yüzlerinden ikisi de "*host-only*" olarak görev yapacaktır. Misafir işletim sistemi ile ilgili ayrıntılar Tablo 2.5'de gösterilmektedir.

Tablo 2.5. Misafir İşletim Sistemi Özellikleri

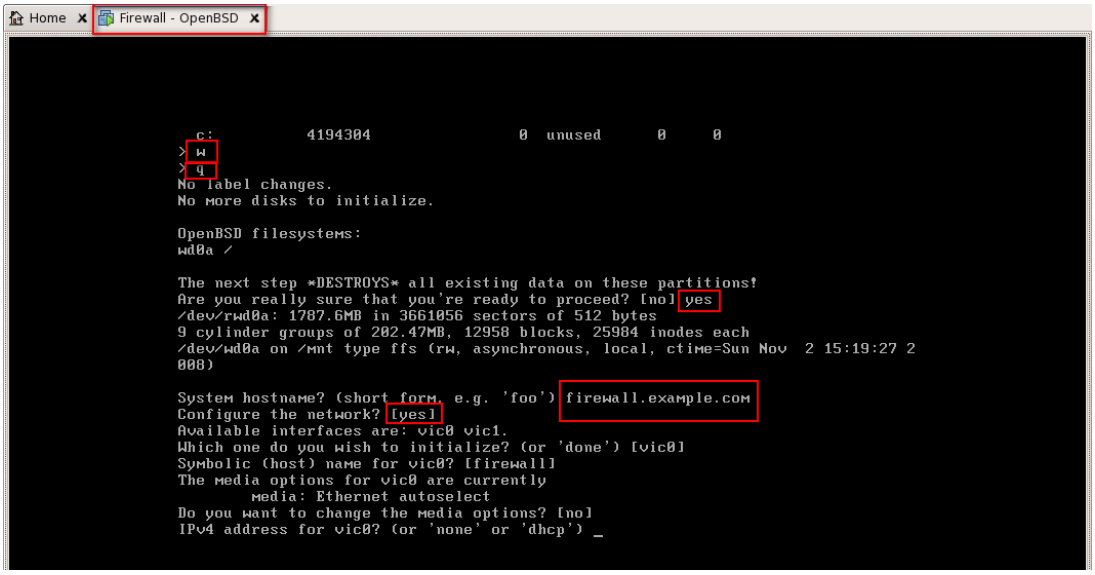
Rolü	Güvenlik Duvarı
İşletim Sistemi	OpenBSD 4.4
Ağ Ara yüzleri	vmnet3 vmnet4
İp Adres/Adres Bloğu	vmnet3 -> 172.16.10.2 vmnet4 -> 172.16.20.2

Vmware Workstation uygulaması içerisinde gerçekleştirilmesi gereken adımların tamamlanmasının ardından, OpenBSD işletim sistemi kurulum adımları başlatılır ve uygun adımları gerçekleştirilerek kurulum başlatılır. Kurulumun esnasında Tcp/Ip yapılandırmasına ilişkin ayarların gerçekleştirildiği ekran görülmektedir. Bu durum Şekil 2.16'da görülmektedir. Makine adı olarak "*firewall.example.com*" kullanılmıştır. Misafir işletim sistemi için kullanılacak olan "*vic0*" ve "*vic1*" ağ ara yüzlerinin yapılandırması için gerekli sorular ekrana gelmektedir. Dış ara yüz olarak kullanılacak "*vic0*", "*vmnet/hub*" olarak kullanılacak ara yüz ise "*vic1*" olmaktadır.

"*vic0*" ağ ara yüzü için burada 172.16.10.2 ip adres bilgisi kullanılmıştır. Eğer ağ üzerinde bulunan bir dhcp (Dynamic Host Configuration Protocol) sunucu üzerinden Tcp/Ip yapılandırması gerçekleştirilecek ise dhcp (Dynamic Host Configuration Protocol) seçilmelidir. Ardından "*vic1*" ağ ara yüzü için belirlenecek ip adres bilgisi ve ön tanımlı ağ geçidi, dns (Domain Name System) sunucu ip adres bilgisi ve "*root*" kullanıcı şifresinin belirleneceği sorulara geçilmektedir. Görüldüğü gibi "*vic1*" ağ ara yüzü için 172.16.20.2 ip adres bilgisi, dns (Domain Name System) sunucu ip adres bilgisi için 172.16.20.3 ip adres bilgisi, ön tanımlı ağ geçidi ip adres bilgisi içinde 172.16.10.1 ip adres bilgisi kullanılmaktadır. Domain name için ise "*example.com*" domaini seçilmiştir. Ayrıca "*root*" kullanıcısı için şifre deneme olarak belirlenmiştir. Tcp/Ip yapılandırmasına ilişkin değerler Tablo 2.6' da topluca gösterilmiştir.

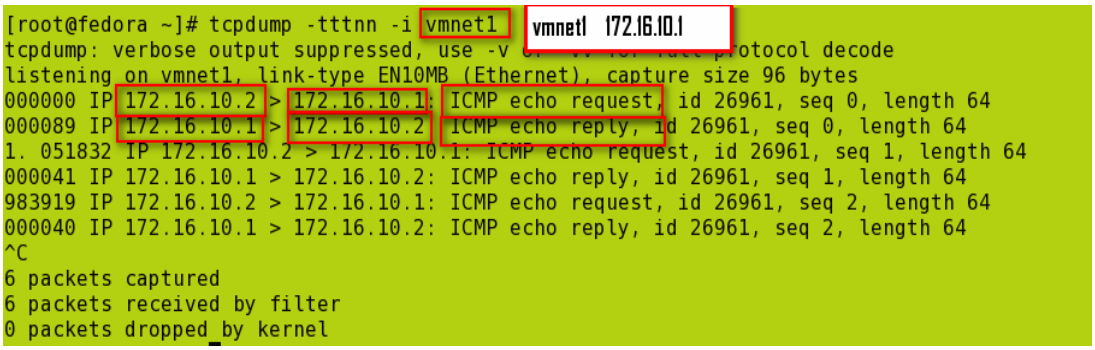
Tablo 2.6. Güvenlik duvarı sunucusu Tcp/Ip yapılandırması

Ağ ara yüzü	**İp Adres Bilgisi**	**Netmask Bilgisi**	**Makine İsmi**	**Ön Tanımlı Ağ Geçidi**
vic0	172.16.10.2	255.255.255.0	firewall.example.com	172.16.10.1
vic1	172.16.20.2	255.255.255.0	firewall.example.com	-



Şekil 2.16. Güvenlik Duvarı Rolündeki Misafir İşletim Sistemi Tcp/Ip Yapılandırması

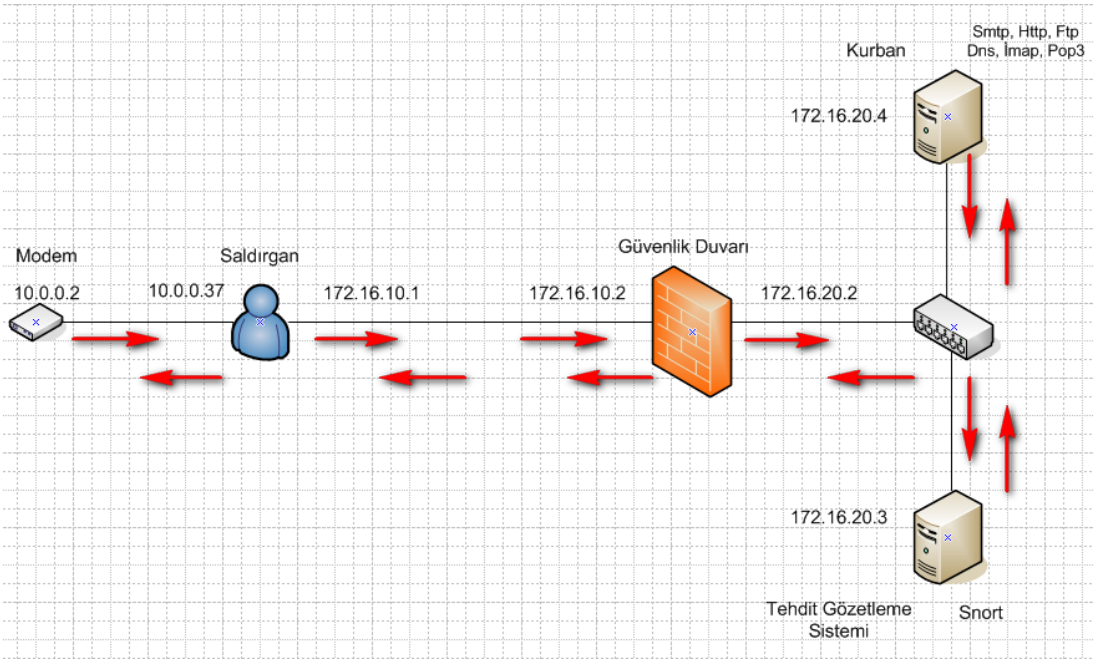
Son olarak güvenlik duvarı rolündeki misafir işletim sistemi üzerinden "*ping*" komutu yardımı ile sunucu sisteme "*icmp echo request*" paketleri gönderilecektir. Sunucu sistem üzerinde Tcp/Ip yapılandırması "*tcpdump*" komutu yardımı ile ilgili paketlerin sistem ulaştığı bilgisi Şekil 2.17'de görüldüğü gibi test edilmiştir.



Şekil 2.17. Güvenlik Duvarı Rolündeki Misafir İşletim Sistemi Tcp/Ip Yapılandırması Doğrulanması

2.4.3. Misafir işletim sistemlerinin internet erişiminin gerçeklenmesi

Şekil 2.18'de gösterilen ağ topolojisi üzerinde kurban ve tehdit gözetleme sistemi misafir işletim sistemlerinin internet erişimini sağlamak için hem güvenlik duvarı hem de sunucu sistem üzerinde NAT işlemi gerçekleştirilmelidir. Burada saldırgan rolündeki sistemin 10.0.0.2 ağ ara yüzü modem ile aynı alt ağ içerisinde bulunmaktadır. Öncelikle güvenlik duvarı (OpenBSD) sisteminde ardından da saldırgan rolündeki sistemde NAT işlemi gerçekleştirilmelidir.



Şekil 2.18. Ağ Topolojisi Üzerinde İp Adres Tanımlamaları

OpenBSD Pf ile misafir işletim sistemlerinin NAT uygulamasını gerçekleştirmek için aşağıdaki adımlar sırası ile uygulanmalıdır. Öncelikle Tcp/Ip paketlerinin ağ kartları arasındaki geçişi sağlamak için "*ip.forwarding*" özelliği aktif hale getirilmelidir. Mevcut durumunu öğrenmek için "*sysctl*" komutu kullanılabilir.

```
# sysctl -a | grep "ip.forwarding"
net.inet.ip.forwarding=0
```

Bu değerin “0” olması aktif olmadığı anlamına gelmektedir. Aktif hale getirmek için yine “*sysctl*” ilgili parametre ile kullanılmalıdır.

```
# sysctl -w net.inet.ip.forwarding=1
net.inet.ip.forwarding: 0 -> 1
# sysctl -a | grep "ip.forwarding"
net.inet.ip.forwarding=1
#
```

Bu değerin sistem açılışında aktif olmasını sağlamak için uygun bir biçimde “*/etc/sysctl.conf*” dosyasının içerisine yazılmalıdır.

```
# echo "net.inet.ip.forwarding=1" >> /etc/sysctl.conf
```

Ardından “*pf*” aktif hale getirilmelidir. Bunun için “*pf*” ile birlikte gelen “*pfctl*” komutu kullanılabilir. Aktif hale getirmek için “*pfctl -e*”, kullanım dışı bırakmak için ise “*pfctl -d*” kullanılmalıdır.

```
# pfctl -e
pf enabled
#
# pfctl -d
pf disabled
```

Sistem açılışında aktif hale getirmek için ise “*/etc/rc.conf*” dosyasının içerisine uygun bir biçimde yazılmalıdır.

```
# cat /etc/rc.conf | grep -E "^pf="
pf=yes                  # Packet filter / NAT
#
```

Güvenlik duvarı rolündeki sistemin ağ yapılandırması aşağıda görüldüğü gibi olmaktadır.

```
# ifconfig
vic0:
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>  mtu
1500
        lladdr 00:0c:29:1d:fa:1c
        groups: egress
        media: Ethernet autoselect
        status: active
        inet 172.16.10.2 netmask 0xffffff00 broadcast 172.16.10.255
        inet6 fe80::20c:29ff:fe1d:fa1c%vic0 prefixlen 64 scopeid 0x1
vic1:
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>  mtu
1500
        lladdr 00:0c:29:1d:fa:26
        media: Ethernet autoselect
        status: active
        inet 172.16.20.2 netmask 0xffffff00 broadcast 172.16.20.255
        inet6 fe80::20c:29ff:fe1d:fa26%vic1 prefixlen 64 scopeid 0x2
enc0: flags=0<> mtu 1536
pflog0: flags=141<UP,RUNNING,PROMISC> mtu 33208
        groups: pflog
#
```

Bu durumda "*vic1*" dış ağ ara yüzü olacaktır. NAT kuralları için gerekli yapılandırma *"/etc/pf.conf*" dosyasına aşağıdaki şekilde yazılmalıdır.

```
# cat /etc/pf.conf
ext_if="vic1"
nat on $ext_if from !($ext_if) -> ($ext_if:0)
#
```

Ardından ilgili kuralların aktif hale gelmesi için "*pfctl -f /etc/pf.conf*" komutu verilmelidir. Artık kurban ve tehdit gözetleme sistemleri güvenlik duvarı rolündeki sistem üzerinden Şekil 2.18'de gösterildiği gibi saldırgan rolündeki sistemin "172.16.10.1" ağ ara yüzüne erişim sağlayabileceklerdir.

Misafir işletim sistemlerinin saldırgan rolündeki sistem üzerinden internet erişimi sağlayabilmesi için saldırgan rolündeki sistem üzerinde de NAT uygulamasının aktifleştirilmesi gerekmektedir. Saldırgan rolündeki sistemin Tcp/Ip yapılandırması aşağıda görüldüğü gibi olmaktadır.

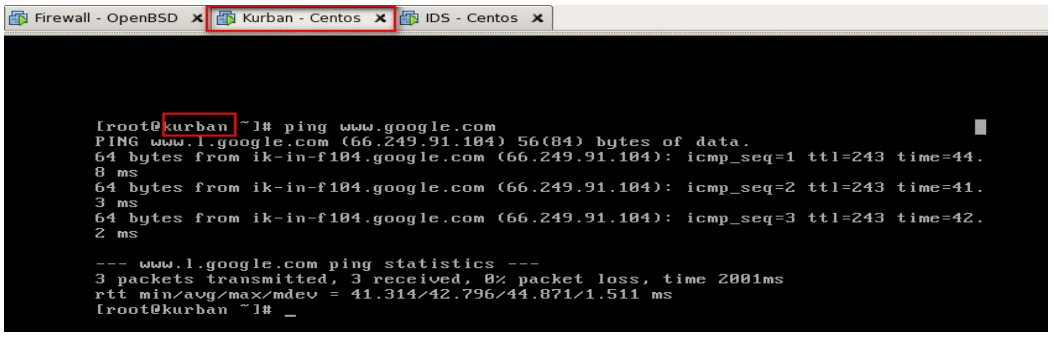
```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:17:A4:DA:17:65
          inet addr:10.0.0.37  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::217:a4ff:feda:1765/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:165628 errors:0 dropped:0 overruns:0 fbelleke:0
          TX packets:114710 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:234621837 (223.7 MiB)  TX bytes:10317098 (9.8 MiB)
          Interrupt:16

vmnet1    Link encap:Ethernet  HWaddr 00:50:56:C0:00:01
          inet addr:172.16.10.1  Bcast:172.16.10.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fec0:1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:64499 errors:0 dropped:0 overruns:0 fbelleke:0
          TX packets:1001 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

vmnet2    Link encap:Ethernet  HWaddr 00:50:56:C0:00:02
          inet addr:172.16.20.1  Bcast:172.16.20.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fec0:2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:315 errors:0 dropped:0 overruns:0 fbelleke:0
```

```
        TX packets:97192 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
#
```

NAT uygulaması ile Tcp/Ip paketlerinin 10.0.0.37 ip adresli ağ ara yüzüne erişimi sağlanmış olacaktır. OpenBSD üzerinde olduğu gibi Tcp/Ip ağ paketlerinin ara yüzler arası geçişini sağlamak için "*ip.forwarding*" özelliğinin aktif hale getirilmelidir ve ardından "*iptables*" için gerekli kurallar uygulanmalıdır. Bunun için aşağıdaki adımlar sırası ile uygulanmalıdır.

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# /sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# /sbin/iptables -A FORWARD -i eth0 -o vmnet1 -m state --state RELATED,ESTABLISHED -j ACCEPT
# /sbin/iptables -A FORWARD -i vmnet1 -o eth0 -j ACCEPT
# /sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Bu işlemlerin gerçekleştirilmesinin ardından, misafir işletim sistemleri internet erişimi sağlanmış olacaktır. Bu durum Şekil 2.19'da gösterilmiştir.



Şekil 2.19 Misafir İşletim Sistemlerinin İnternet Erişiminin Doğrulanması

2.5. Tehdit gözetleme sistemi rolündeki misafir işletim sistemi için gerekli sunucu servislerinin aktif hale getirilmesi

Tehdit gözetleme sistemi rolündeki misafir işletim sistemi üzerinde Tablo 2.7'de gösterildiği gibidir.

Tablo 2.7. Misafir İşletim Sistemi, Sunucu Servisi Ve Sistem Üzerinde Çalışan Sunucu Servis Yazılımları

Misafir İşletim Sistemi	**Sunucu Servisi**	**Sistem Üzerinde Çalışan Sunucu Servis Yazılımı**
Tehdit Gözetleme Sistemi	Tehdit Gözetleme Servisi	Snort
	Veritabanı Sunucusu	MySQL
	Tehdit Gözetleme Sistemi Web Erişim Ara yüzü	Base
	Web	Apache
	Betik Dili	PHP

Tehdit gözetleme sistemi makinesi üzerinde tehdit gözetleme sistemi olarak Snort kurulumu gerçekleştirilecektir. Snort işlem kayıt bilgilerini tutmak için MySQL veritabanı sunucusu kullanılacaktır. Bu işlem kayıt bilgilerinin web arabiriminden izlenebilmesi için "*acid*" ve "*base*" yazılımları kullanılacaktır. Web sunucu yazılımı olarak ise Apache kullanılacaktır.

2.5.1. Snort tehdit gözetleme servisinin kurulumu ve yapılandırılması

Snort kurulumu için öncelikle gerekli kaynak kodun temin edilmesi gerekmektedir. Kurulum öncesinde Snort kurulumu ve devamında işlem kayıt bilgilerinin takip edilmesi için web arabiriminin düzenlenmesi için gerekli paketler "*yum*" paket yönetim sistemi yardımı ile gerçekleştirilecektir.

```
# yum -y install mysql-server libpcap-devel pcre pcre-devel mysql-devel
# chkconfig mysqld on
```

İlgili paketlerin kurulumu ve gerekli yapılandırmanın gerçekleştirilmesinin ardından Snort için kullanılacak en kararlı sürümü temin edilmelidir. Kaynak koddan kurulum gerçekleştirileceği "*gcc*"'nin (Gnu Compiler Collection) sistemde kurulu olması gerekmektedir.

```
# yum install gcc
# cd /tmp
# wget http://www.snort.org/dl/snort-2.8.3.2.tar.gz
# tar -zxvf snort-2.8.3.2.tar.gz
# cd snort-2.8.3.2
# ./configure --prefix=/usr/local/snort --with-MySQL --enable-dynamicplugin --with-MySQL-includes=/usr/include/MySQL/
# make
# make install
```

Kurulumun tamamlanmasın ardından Snort servisinin çalışması için gerekli grup/kullanıcı ikilisi oluşturulmalı, yapılandırma için kullanılacak olan dosyalar Snort kaynak dizininden "*--prefix*" parametresi ile belirtilen dizine kopyalanmalıdır.

```
# groupadd snort
# adduser –c "Snort User" –s /sbin/nologin –g snort snort
# mkdir /usr/local/snort/{etc,rules}
```

```
# cp /tmp/snort-2.8.3.2/etc/*  /usr/local/snort/etc
```

Snort yapılandırması için aşağıdaki adımlar topolojiye göre uygulanmalıdır. Snort işlem kayıt bilgilerinin tutulması için MySQL veritabanı kullanılacaktır.

```
# cd /usr/local/snort/etc
# vi snort.conf
  var HOME_NET 172.16.20.0/24
  var EXTERNAL_NET !HOME_NET
  dynamicpreprocessor directory
/usr/local/snort/snort_dynamicpreprocessor/
  output database: log, MySQL, user=snort password=sifre dbname=snort
  host=127.0.0.1
#
```

Snort işlem kayıt bilgilerinin MySQL veritabanında tutulabilmesi için gerekli veritabanı ve veritabanı erişimi için MySQL "*root*" kullanıcısı harici bir MySQL kullanıcısı, bu veritabanı üzerinde bütün erişim izinlerine sahip olacak şekilde oluşturulmalıdır. Bunun için aşağıdaki adımlar sırası ile uygulanmalıdır.

```
# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.0.45 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> create database snort;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL ON snort.* TO snort@localhost IDENTIFIED BY
'sifre';
Query OK, 0 rows affected (0.01 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> quit
```

Gerekli tabloların oluşturulması için Snort kaynak dizininde bulunan yardımcı betik kullanılabilir. Bunun için aşağıdaki adımlar sırası ile uygulanmalıdır.

```
# cd /tmp/snort-2.8.3.2/schemas
# mysql -u snort -psifre < create_mysql snort
```

Snort kuralları bir kaç şekilde temin edilebilmektedir. Burada kayıtlı kullanıcı olmadan gerekli kurallar temin edilecektir. Bunun için aşağıdaki adımlar sırası ile uygulanmalıdır.

```
# cd /usr/local/snort/rules
#wget http://www.snort.org/pubbin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.4.tar.gz
```

Snort aşağıda belirtilen parametreler ile başlatılmalıdır.

```
# mkdir /var/log/snort
# /usr/local/snort/bin/snort -i eth0 -D -c /usr/local/snort/etc/snort.conf --dynamic-preprocessor-lib-dir=/usr/local/snort/lib/snort_dynamicpreprocessor/
```

2.5.2. Snort servisi için kullanılabilecek başlatma durdurma betiği ve sistem açılışında aktif hale gelmesi

Snort servisi için başlatma ve durdurma için kullanılabilecek olan betik aşağıda gösterilmiştir.

```
#!/bin/sh
#
# Snort Start Stop Script
#
```

```
# chkconfig: 345 80 20
# description: Control Snort

case "$1" in
'start')
/usr/local/snort/bin/snort -i eth0 -D -c /usr/local/snort/etc/snort.conf --
dynamic-preprocessor-lib-
dir=/usr/local/snort/lib/snort_dynamicpreprocessor/ &
;;
'stop')
kill -9 `ps -ef | grep "snort" | grep -v "grep" | awk '{print $2}'`
;;
*)
echo "Usage: $0 { start | stop }"
exit 1
;;
esac
exit 0
#
```

Servisi başlatmak ve durdurmak için "*start*" ve "*stop*" parametrelerinin kullanılması yeterlidir. Örnek bir kullanım aşağıda gösterilmiştir.

```
# /etc/init.d/snort start
```

Yukarıda gösterilen betik "*/etc/init.d/snort*" olarak kaydedildikten sonra gerekli izinler verilmelidir. Bunun için aşağıdaki adımlar sırası ile uygulanmalıdır.

```
# chown root:root /etc/init.d/snort
# chmod 755 /etc/init.d/snort
# ln -s /etc/init.d/snort /etc/rc0.d/K60snort
# ln -s /etc/init.d/snort /etc/rc3.d/S80snort
# chkconfig snort on
```

2.5.3. Apache, php kurulumu ve yapılandırması

Snort işlem kayıt bilgilerinin MySQL veritabanında sorgulanıp web arabiriminden rahatça izlenebilmesi için "*base*" kurulumu gerçekleştirilecektir. Bunun için öncelikle Apache web sunucu yazılımı ve "*php*" kurulumu gerçekleştirilmelidir. Bunun için aşağıdaki adımlar sırası ile uygulanmalıdır.

```
# yum -y install httpd php php-mysql php-gd
# chkconfig httpd on
# /etc/init.d/httpd start
```

2.5.4. Base kurulumu ve yapılandırması

"*Base*" kurulumu için gerekli olan "*adodb*" aşağıda belirtilen şekilde yapılandırılmalıdır.

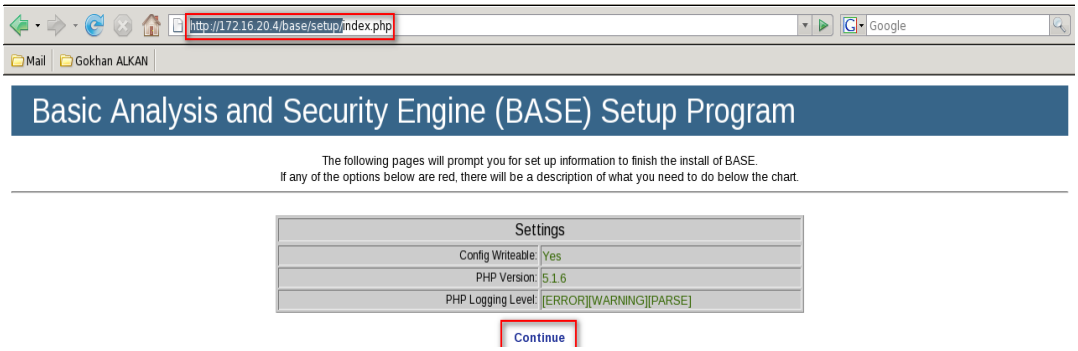
```
# wget http://kent.dl.sourceforge.net/sourceforge/adodb/adodb506a.tgz
# tar -zxvf adodb506a.tgz
# mv adodb5 /var/www/html/adodb
```

Ardından "*base*" kurulumu için gerekli adımlar aşağıda belirtildiği şekilde takip edilmelidir. Öncelikle kaynak kod temin edilmeli ve ardından web arabiriminden kurulum adımları izlenmelidir.

```
# wget http://kent.dl.sourceforge.net/sourceforge/secureideas/base-
1.4.1.tar.gz
# tar -zxvf base-1.4.1.tar.gz
# cp -pR base /var/www/html/
# vi /etc/php.ini
...
   error_reporting  =  E_ALL & ~E_NOTICE
```

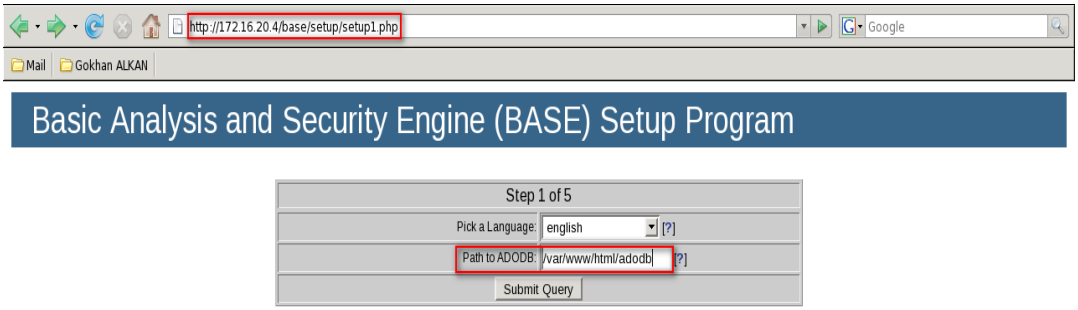
```
...
#
# chown -R root:apache base
# chmod 570 base/
```

Yukarıda gösterilen ilgili ayarların gerçekleştirilmesinin ardından web tarayıcı adres satırına "*http://172.16.20.4/base/setup/*" yazılarak kurulum adımları başlatılmalıdır.



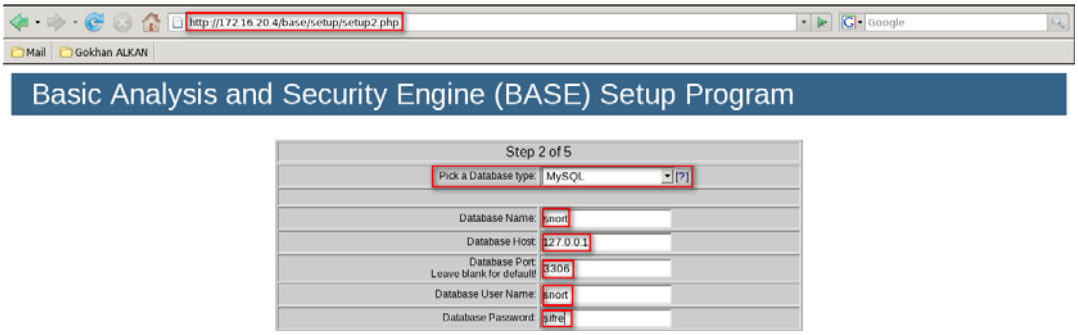
Şekil 2.20. Base Yapılandırma Adımı - 1

Şekil 2.20'de görüldüğü gibi Apache ve "*php*" yapılandırması kontrol edilmekte ve eğer düzeltilmesi gereken hata mesajı varsa ekrana uyarı olarak gösterilmektedir. Eğer hata mesajı alınmıyorsa "*base*" kurulum adımları devam etmektedir.



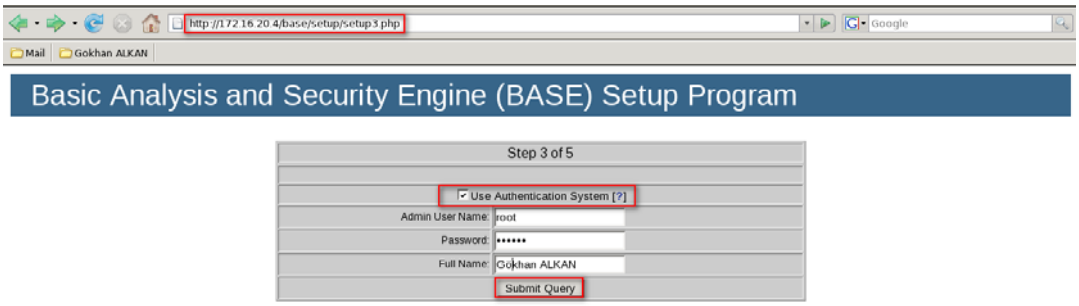
Şekil 2.21. Base Yapılandırma Adımı - 2

Bir sonraki adımda ise "*adodb*" tam yolu gösterilmelidir. Kurulum adımlarında "*adodb*" tam yolu olarak "*/var/www/html/adodb*" seçilmiştir. Bu durum Şekil 2.21' de gösterilmektedir.



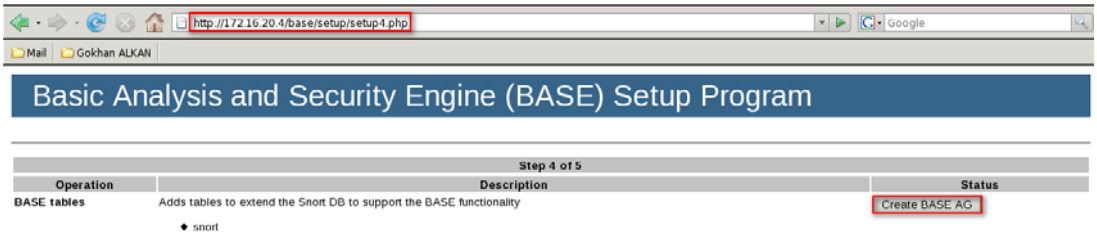
Şekil 2.22. Base Yapılandırma Adımı - 3

Bir sonraki adım Snort saldırı tespit sisteminin işlem bilgileri tutmak için kullandığı MySQL veritabanı sunucusuna ait yapılandırma bilgilerinin belirtileceği bölümdür. Kurulum adımlarında veritabanı adı olarak Snort, veritabanı kullanıcısı olarak snort, snort veritabanı kullanıcısının parola bilgisi olarak snort, veritabanı sunucusunun çalıştığı makine adı olarak 127.0.0.1 ve son olarak MySQL veritabanı ön tanımlı çalışma portu olarak da 3306 seçilmiştir. Bu değerlere göre Şekil 2.22'de gerekli yapılandırma gerçekleştirilmiştir.



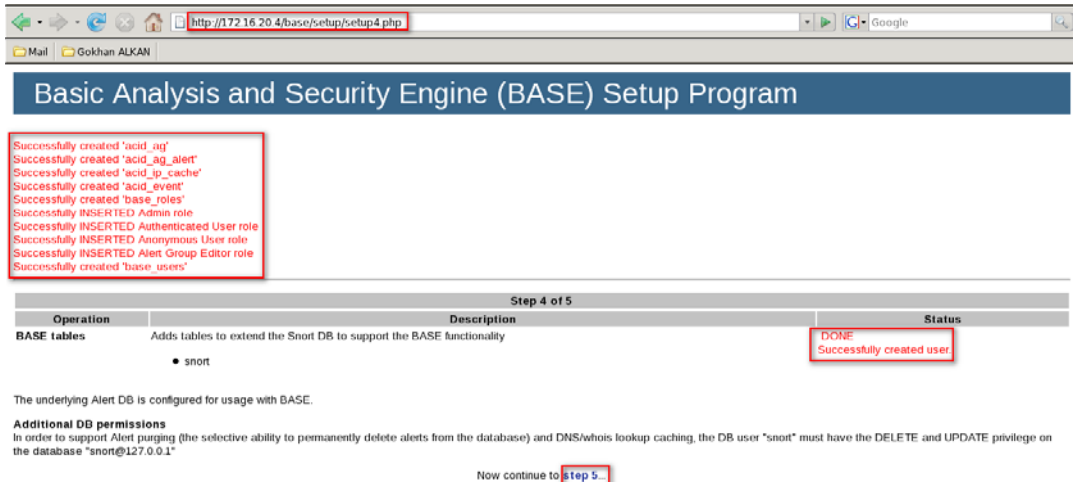
Şekil 2.23. Base Yapılandırma Adımı - 4

Bir sonraki adımda "*base*" kullanımının kullanıcı adı parola kullanılarak gerçekleştirilmesi için gerekli adımlar bulunmaktadır. Şekil 2.23'de görüldüğü gibi "*base*" servisi yardımı ile web arabirimini kullanabilmek için kullanıcı adı ve parola yetkilendirmesi gerçekleştirilmiştir. Burada kullanıcı adı olarak "*root*" parola bilgisi olarak ise deneme seçilmiştir.



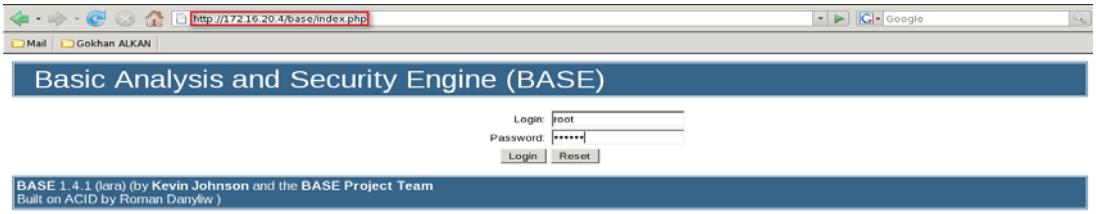
Şekil 2.24. Base Yapılandırma Adımı - 5

"*Base*" için oluşturulacak olan tablolar için "*Create BASE AG*" butonu kullanılmalıdır. Bu durum Şekil 2.24'de gösterilmektedir.



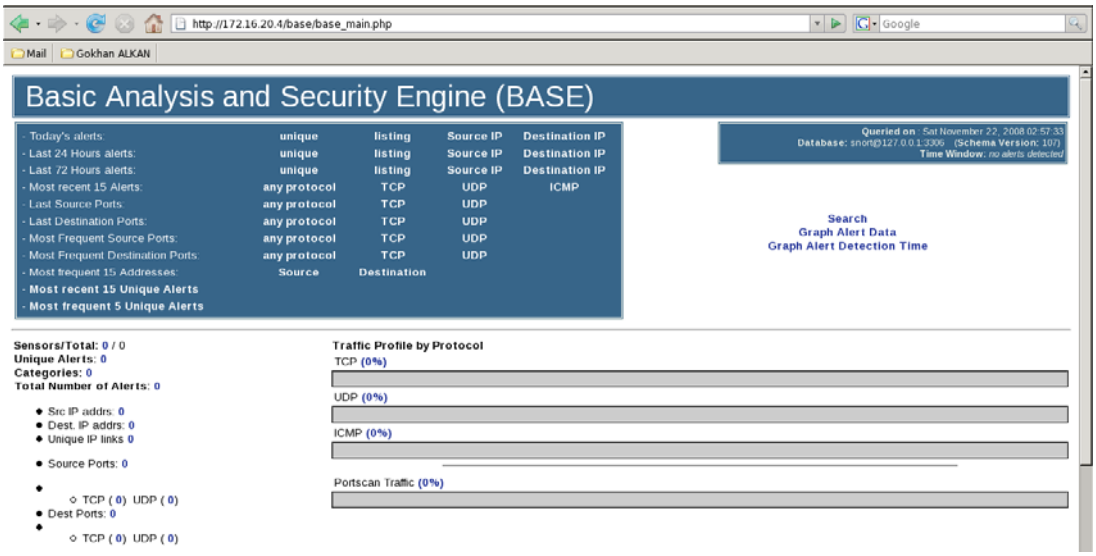
Şekil 2.25. Base Yapılandırma Adımı - 6

Bir diğer adımda kurulumun sorunsuz gerçekleştiğine dair durum mesajı Şekil 2.25'de görülmektedir.



Şekil 2.26. Base Yapılandırma Adımı - 7

Base web arabirimine giriş gerçekleştirmek için kurulum esnasında belirtilen kullanıcı adı ve parola bilgileri kullanılarak giriş gerçekleştirilmektedir. Bu durum Şekil 2.26'de gösterilmektedir.



Şekil 2.27. Base Yapılandırma Adımı - 8

Kurulumun tamamlanmasının ardından "*base*" web arabirimi Şekil 2.27'de görüldüğü gibi olmaktadır.

İşletim sistemi, sunucu servislerinde kullanılan kullanıcı isimleri ve kullanılan adresler işletim sistemi rollerine Tablo 2.8 ve Tablo 2.9' dan ulaşılabilir.

Tablo 2.8. İşletim Sistemi ve Sunucu Servislerinde Kullanılan Kullanıcı Adları

Kullanıcı Adı	**İşletim Sistemi**	**İşletim Sistemi Rolü**	**Servis Adı**
root	Linux	Sunucu	
galkan	Linux	Sunucu	
snort	Linux	Tehdit Gözetleme Sistemi	Mysql
root	Linux	Tehdit Gözetleme Sistemi	Mysql
galkan	Linux	Kurban	

Tablo 2.9. Kullanılan Adresler ve İşletim Sistemi Rolleri

Adres	**İşletim Sistemi Rolü**
kurban.example.com	Kurban
ids.example.com	Tehdit Gözetleme Sistemi
firewall.example.com	Güvenlik Duvarı

BÖLÜM 3. SCAPY YAZILIM KÜTÜPHANESİ

Scapy yazılım kütüphanesi python ile bütünleşik olarak kullanılan Tcp/Ip protokol kümesine ait paket oluşturma, gönderme, alma gibi işlemleri gerçekleştirmek için kullanılan bir yazılım kütüphanesidir. Windows ve Linux işletim sistemleri için kullanımı mevcuttur. Kullanılan Linux dağıtımları için de kurulum yönergeleri farklılık göstermektedir. Kurulum ve kullanım detayları ilgili başlıklar altında gösterilmiştir.

3.1. Scapy Yazılım Kütüphanesinin Kurulumu ve Python ile Kullanımı

Fedora linux dağıtımı için scapy yazılım kütüphanesinin kurulumu ve bütünleşik olarak python ile kullanımı için gerekli paketlerin sisteme yüklenmesi gerekmektedir. Burada sadece Fedora linux dağıtımına özel kurulum adımları gerçekleştirilmiştir. Bu adımlar diğer dağıtımlar içinde uygulanabilir olmaktadır. Bunun için aşağıdaki adımların sırası ile uygulanması gerekmektedir.

```
# yum install mercurial python-devel
# cd /tmp
# hg clone http://hg.secdev.org/scapy
# cd scapy
# python setup.py install
```

Scapy yazılım kütüphanesinin ekstra özelliklerinin kullanılabilmesi için aşağıda belirtilen paketlerinde kurulması gerekmektedir. Bu paketlerin kurulması zorunluluk olmamak ile birlikte ekstra özelliklerin kullanılması için gereklidir. Bunun için aşağıdaki adımlar sırası ile takip edilmelidir.

```
# yum install graphviz python-crypto sox PyX gnuplot numpy
```

```
# cd /tmp
# wget http://heanet.dl.sourceforge.net/sourceforge/gnuplot-py/gnuplot-py-1.8.tar.gz
# tar xvfz gnuplot-py-1.8.tar.gz
# cd gnuplot-py-1.8
# python setup.py install
```

Scapy yazılım kütüphanesinin kurulumunun ardından deneme amaçlı örnek bir kod parçası çalıştırılarak kütüphanenin sağlıklı bir biçimde çalıştığı test edilmektedir.

```
#!/usr/bin/python
import sys
from scapy.all import sr1,IP,ICMP
p=sr1(IP(dst=sys.argv[1])/ICMP())
if p:
    p.show()
```

Belirtilen kod parçası "*deneme_scapy.py*" olarak kaydedilir ve ardından gerekli izinler verilerek çalıştırılmalıdır. Bunun için aşağıdaki adımlar sırası ile uygulanmalıdır.

```
# chmod 755 scapy_deneme.py
# ./scapy_deneme.py 10.0.0.2
###[ IP ]###
  version= 4L
  ihl= 5L
  tos= 0x0
  len= 28
  id= 64276
  flags=
  frag= 0L
  ttl= 64
  proto= icmp
  chksum= 0x6ba6
  src= 10.0.0.2
```

```
  dst= 10.0.0.37
  options= ''
###[ ICMP ]###
     type= echo-reply
     code= 0
     chksum= 0xffff
     id= 0x0
     seq= 0x0
###[ Padding ]###
        load=
'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00
```

Oluşturulan "*icmp echo request*" paketi için 10.0.0.2 ip adresli sistemden gelen cevap görülmektedir.

3.2. Scapy Yazılım Kütüphanesi Kullanımı

Scapy, bu çalışmada Tcp/Ip protocol kümesine ait paket oluşturma, paket gönderme ve paket alma gibi işlemleri gerçekleştirmek için python ile bütünleşik olarak kullanılmıştır.

3.2.1. Scapy yazılım kütüphanesi yardım fonksiyonları

Scapy yazılım kütüphanesi için kullanılabilecek 2 önemli yardım fonksiyonu vardır. Genel olarak "*lsc*" ile mevcut fonksiyonları, "*ls*" ile de istenilen bir fonksiyona ait özellikler görüntülenebilir. Bu anlatımı destekleyecek bir kullanım aşağıda gösterilmiştir.

```
>>> lsc()
arpcachepoison          : Poison target's cache with (your MAC,victim's IP) couple
arping                  : Send ARP who-has requests to determine which hosts are up
…
…
tshark                  : Sniff packets and print them calling pkt.show(), a bit like text wireshark
wireshark               : Run wireshark on a list of packets
wrpcap                  : Write a list of packets to a pcap file
```

"*ls*" fonksiyonu yardımı ile de istenen bir scapy fonksiyonu hakkında ayrıntılı bilgi edinilmektedir. Bu duruma örnek bir kullanım aşağıda gösterilmiştir.

```
>>>pkt=IP()
>>>ls(IP)
version   :              BitField                =                 (4)
ihl       :              BitField                =                 (None)
tos       :              XByteField              =                 (0)
len       :              ShortField              =                 (None)
id        :              ShortField              =                 (1)
flags     :              FlagsField              =                 (0)
frag      :              BitField                =                 (0)
ttl       :           ByteField                  =                 (64)
proto     :              ByteEnumField                  =          (0)
chksum    :                 XShortField          =                 (None)
src       :              Emph                    =                 (None)
dst       :              Emph                    =                 ('127.0.0.1')
options   :                 PacketListField      =                 ([])
```

"*lsc*" ve "*ls*" fonksiyonları yardımı ile çevrim dışı olarak kullanılabilecek olan değerler hakkında bilgi edinilebilir.

Ip ve tcp katmanında paket oluşturmak, oluşturulan paketin kaynak ip adres bilgisine istenen bir ip adres bilgisini atamak ve ardından etkileşimli modda görüntülemeye ilişkin örnek bir kullanım aşağıda gösterilmiştir.

```
>>>p=IP(src="172.16.10.1")
>>>p.src
'172.16.10.1'
>>>p=IP(src="172.16.10.1")/TCP()
>>>ls(p)
version    :        BitField                =          4                    (4)
ihl        :        BitField                =          None                 (None)
tos        :        XByteField              =          0                    (0)
len        :        ShortField              =          None                 (None)
id         :        ShortField              =          1                    (1)
flags      :        FlagsField              =          0                    (0)
frag       :        BitField                =          0                    (0)
ttl        :    ByteField                   =          64                   (64)
proto      :        ByteEnumField           =          6                    (0)
chksum     :             XShortField        =          None                 (None)
src        :        Emph                    =          '172.16.10.1'        (None)
dst        :        Emph                    =          '127.0.0.1'          ('127.0.0.1')
options    :        PacketListField         =          []                   ([])
--
sport      :        ShortEnumField          =          20                   (20)
dport      :        ShortEnumField          =          80                   (80)
seq        :        IntField                =          0                    (0)
ack        :        IntField                =          0                    (0)
dataofs    :        BitField                =          None                 (None)
reserved   :             BitField          =          0                    (0)
flags      :        FlagsField              =          2                    (2)
window     :             ShortField         =          8192                 (8192)
chksum     :             XShortField        =          None                 (None)
urgptr     :        ShortField              =          0                    (0)
options    :             TCPOptionsField    =          {}
         ({})
```

Aynı şekilde oluşturulan paketin içerik (payload) kısmına istenen bir verinin yazılması durumu için de "*raw*" katman bilgisi eklenmelidir.

```
>>>p=IP(src="172.16.10.1")/TCP()/Raw("load")
>>>ls(p)
version   :          BitField          =              4                    (4)
ihl       :          BitField          =            None                   (None)
tos       :          XByteField        =              0                    (0)
len       :          ShortField        =            None                   (None)
id        :          ShortField        =              1                    (1)
flags     :          FlagsField        =              0                    (0)
frag      :    BitField          =              0                    (0)
ttl       :    ByteField         =             64                    (64)
proto     :    ByteEnumField=                   6                    (0)
chksum:          XShortField       =            None                   (None)
src       :          Emph              =        '172.16.10.1'              (None)
dst       :          Emph              =        '127.0.0.1'          ('127.0.0.1')
options:         PacketListField=               []                   ([])
--
sport     :          ShortEnumField          =        20                 (20)
dport     :          ShortEnumField          =        80                 (80)
seq       :          IntField          =              0                  (0)
ack       :          IntField          =              0                  (0)
dataofs   :          BitField          =            None                 (None)
reserved:          BitField          =              0                  (0)
flags     :          FlagsField        =              2                  (2)
window:          ShortField        =            8192                 (8192)
chksum    :          XShortField       =            None                 (None)
urgptr    :          ShortField        =              0                  (0)
options   :          TCPOptionsField   =              {}                 ({})
--
load      : StrField             =                 'load'
          (")
```

3.2.2. Scapy yazılım kütüphanesi kullanım modları

Scapy her ne kadar python ile bütünleşik biçimde kullanılabilecek olan bir Tcp/Ip yazılım kütüphanesi olarak kullanılabilse de bunun yanında interaktif olarak kendi başına da kullanılabilmektedir. İnteraktif şekilde sadece paket detaylarına odaklanarak gerekli kontrolleri yapmak mümkün olmaktadır. İnteraktif kullanıma örnek aşağıda gösterildiği gibi olmaktadır.

```
# python
>>> from scapy.all import *
>>> p=IP(dst="10.0.0.2")/ICMP()
>>> sr1(p)
<IP  version=4L ihl=5L tos=0x0 len=28 id=6602 flags= frag=0L ttl=64 proto=icmp chksum=0x4cf1 src=10.0.0.2 dst=10.0.0.37 options='' |<ICMP type=echo-reply code=0 chksum=0xffff id=0x0 seq=0x0 |<Padding load='\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00' |>>>
>>>
```

Sürüm 2.x ile birlikte kullanımı için "*scapy.all import* *" kullanılmalıdır. Sürüm 1.x ile birlikte "*scapy import* *." kullanılmalıdır.

Python ile bütünleşik kullanımına örnek bir kod bloğu ise aşağıda gösterildiği gibi olmaktadır.

```
#!/usr/bin/python
from scapy.all import *
p=IP(dst="10.0.0.2")/ICMP()
sr1(p)
```

3.2.3. Scapy yazılım kütüphanesi ve TCP/IP katmanları

Scapy yazılım kütüphanesi ile paket oluşturmak için kullanılan temel Tcp/Ip katman yapısı aşağıda gösterildiği gibidir. Oluşturulmak istenen paket Tablo 3.1'de gösterilen katmanlardan hepsini ya da bazılarını içerebilir. Burada belirtilen her katman Tcp/Ip seviyesinde bir katmana karşılık gelmektedir. Oluşturulmak istenen paket de bulunması gereken scapy yazılım kütüphanesi katmanları seçilerek gerekli paket oluşturulmalıdır.

Tablo 3.1. Scapy ve Temel Tcp/Ip Katmanları

PADDING
PAYLOAD
ICMP/UDP/TCP
IP
ARP

192.168.1.3 ip adres bilgisine sahip hedef sistemin 80 tcp portuna "*syn*" bayrağı aktif hale getirilmiş bir tcp paketini, scapy yazılım kütüphanesi ile oluşturmak için aşağıda gösterilen kod bloğu kullanılabilir.

```
p=IP(dst="192.168.1.3")/TCP(dport=80,flags="S")
```

3.2.4. Paket gönderme

Scapy yazılım kütüphanesi ile paket göndermek için kullanılabilecek olan birden fazla fonksiyon bulunmaktadır. Oluşturulmak istenen paket ister katman 2'de ister katman 3'de gönderilebilmektedir. Temek olarak scapy yazılım kütüphanesi ile paket

oluşturmak için kullanılabilecek olan fonksiyonlar "*sr*", "*sr1*", "*srp*" ve "*srp1*" olarak göze batmaktadır. İlgili fonksiyonlar Tablo 3.2'de gösterilmiştir.

Tablo 3.2. Scapy Yazılım Kütüphanesi Temel Paket Gönderme Fonksiyonları

Fonksiyon Adı	**Fonksiyon İşlevi**
sr	Paketleri 3. katmanda gönderir ve alır.
sr1	Paketleri 3. katmanda gönderir sadece 1. cevabı alır.
srp	Paketleri 2. katmanda gönderir ve alır.
srp1	Paketleri 2. katmanda gönderir ve sadece 1. cevabı alır.

"*sr*" fonksiyonlarında (srp, srp1 vb) p ile gösterilen bölüm "*PF_PACKET*", Linux çekirdeğinin 2. katmanda paket göndermesine izin veren ara yüzden gelmektedir.

192.168.1.3 ip adres bilgisine sahip hedef sistemin 22 tcp portuna "*syn*" bayrağı aktif hale getirilmiş bir tcp paketini scapy ile oluşturmak için aşağıda gösterilen kod bloğu kullanılabilir.

```
...
p=IP(dst="192.168.1.3")/TCP(dport=22,flags="S")
sr1(p)
...
```

3.2.5. Paket gönderip alma

Tcp/Ip paketleri oluşturmak, göndermek ve gelen cevabı almak için genel olarak "*sr*" fonksiyonu kullanılmaktadır. Hedef sistemin tcp/80 en bilinen port numaralarını tarayarak, açık portları tespit eden örnek bir kod bloğu için;

```
#!/usr/bin/env python

from scapy.all import *
import sys,re

ip_regexp=re.compile("^(\d{,3})\.(\d{,3})\.(\d{,3})\.(\d{,3})$")
target=sys.argv[1]

if not len(sys.argv[1]) == 2:
   if re.match(ip_regexp,target):

pkt=IP(src="172.16.10.1",dst=target)/TCP(sport=RandShort(),dport=[(1,102
4)],flags='S')
       ans,unans=sr(pkt,timeout=0.1)

       print "Acik Portlar\n------------------------"
       for s,r in ans:
          print r[IP].src,"-->",r[TCP].sport
   else:
       print "Hedef Sistem Gecerli Bir Ip Adresine Sahip Olmali !!!"
       sys.exit(1)
else:
   print "Hedef Sistem Icin Bir Ip Adres Bilgisine Sahip Olmali !!!"
   sys.exit(2)
```

Örnek kod parçası çalıştırıldığında çıktısı aşağıda görüldüğü gibi olmaktadır.

```
# ./port_scan.py 172.16.20.4
Acik Portlar
------------------------
172.16.20.4 --> 22
172.16.20.4 --> 80
172.16.20.4 --> 110
172.16.20.4 --> 143
172.16.20.4 --> 443
172.16.20.4 --> 993
172.16.20.4 --> 995
```

3.2.6. Paket koklama (Sniffing)

Scapy yazılım kütüphanesi ile paket yakalamak için "*sniff*" fonksiyonu kullanılmaktadır.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import re
from scapy.all import *

def sniffPackets():
    sniff(store=1,filter="tcp or udp or icmp",iface="vmnet3",prn = lambda x:
show_packet(x))

def show_packet(packet):
    proto=packet.sprintf("%IP.proto%")
    print packet.sprintf("%IP.proto% -> %IP.src% -> %IP.dst%")

### Main
if__name__=="__main__":
   sniffPackets()
```

```
# ./fidsd.py 172.16.10.1
icmp -> 172.16.10.2 -> 172.16.10.1
icmp -> 172.16.10.1 -> 172.16.10.2
icmp -> 172.16.10.2 -> 172.16.10.1
```

BÖLÜM 4. UYGULAMANIN GERÇEKLENMESİ

Güvenlik duvarı ve tehdit gözetleme sistemlerini otomatize olarak test edilmesi işlemini gerçekleştiren ”*fidstester*”, ”*fidsd*” ve ”*fidsreport*” yazılımlarının kullanım parametreleri ve örnek kullanımları ilgili başlıklar altında detaylı olarak incelenmiştir.

4.1. Uygulama Ait Genel Tanımlamalar ve Kullanım Paremetreleri

”*Fidstester*” python programlama dili ile Linux sistemlerde üzerinde ve scapy yazılım kütüphanesi kullanılarak, güvenlik duvarı ve tehdit gözetleme sistemlerinin otomatize test edilmesi için geliştirilmiş bir uygulamadır.

”*Fidstester*” uygulaması, 3 yazılım parçasından oluşmaktadır. Bunlar genel tanımlamaları ile; belirtilen yapılandırma dosyasına göre paket oluşturup paketleri gönderen ”*fidstester*”, paket dinleyen ”*fidsd*” ve son olarak her iki yazılımın oluşturduğu kayıt bilgilerini karşılaştırıp güvenlik duvarı kurallarını listeleyen ”*fidsreport*” yazılımlarıdır.

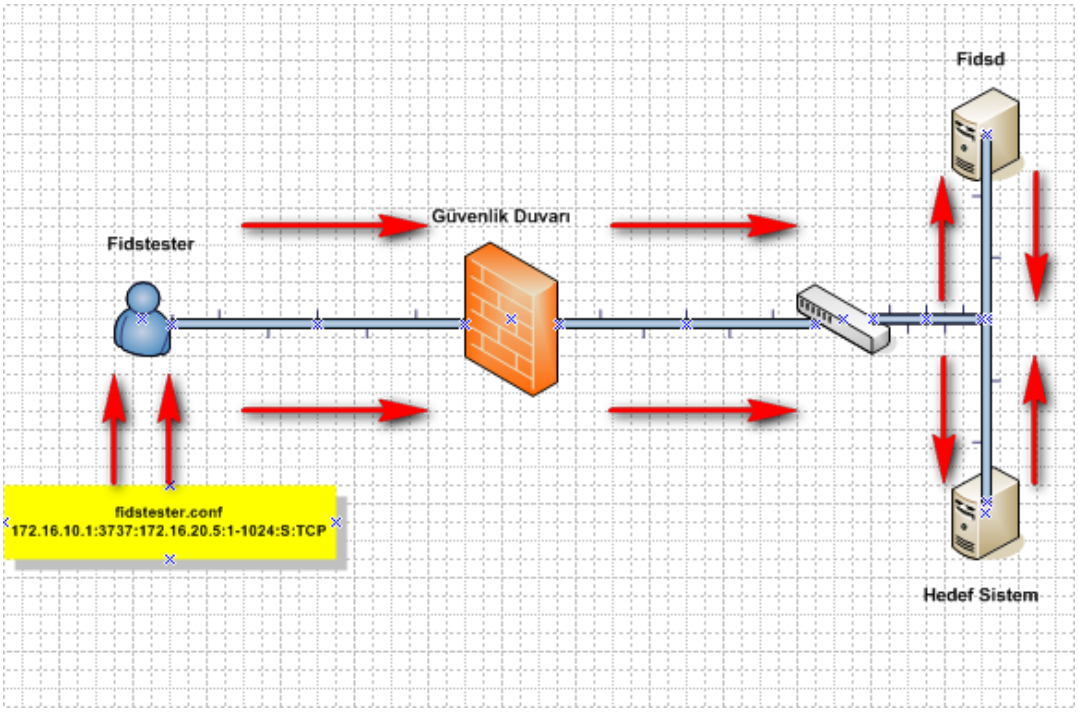
4.2. Fidstester

”*Fidstester*” belirtilen yapılandırma dosyasına göre ilgili paketleri oluşturarak hedef sisteme göndermektedir. Hedef sisteme gönderilen her paket çalışma zamanı esnasında ”*fidstester.log*” kayıt dosyasına işlenerek daha sonra ”*fidsreport*” yazılımı tarafından kullanılmak üzere çalışma dizininde oluşturulmaktadır. Kayıt işlemleri için kullanılan dosyanın adı ”*fidstester.log*” olacaktır. Bu durum Şekil 4.1’de gösterilmektedir.

Yapılandırma dosyasının genel formatı *":"* ile birbirinden ayrılmış alanlardan oluşmaktadır. Her bir bölümde ilgili paketin oluşturulması için gerekli Tcp/Ip alanları bulunmaktadır. Örnek bir alan formatı aşağıda gösterilmektedir.

```
172.16.10.1:3737:172.16.20.5:80:S:TCP
```

Bu kullanıma göre 172.16.10.1 kaynak ip adres bilgisine ve 3737 kaynak port numarasına sahip TCP paketi, 172.16.20.5 ip adres bilgisine sahip hedef sistemin 80 numaralı portuna gönderilecektir.



Şekil 4.1. Fidstester Çalışma Şeması

4.2.1. Fidstester kullanım paremetreleri

”*Fidstester*” uygulamasına ait kullanım paremetreleri ve bu paremetrelere ait açıklamalar aşağıda gösterildiği gibi olmaktadır.

[-f] yapılandırma_dosyası : Belirtilen yapılandırma dosyasına göre ilgili paketlerin oluşturulacağı dosyadır. Aynı şekilde yapılandırma dosyasına göre oluşturulup gönderilen her paket daha sonra ”*fidsreport*” uygulaması tarafından kullanılmak üzere ”*fidstester.log*” isminde kayıt altına alınacaktır. Yapılandırma dosyasının belirtimi zorunludur. Örnek bir kullanımı gösterildiği gibi olmaktadır.

```
# fidstester -f fidstester.conf
```

[-i] paketlerin_gönderileceği_arayüz : Oluşturulan paketlerin eğer sistem üzerinde birden fazla ağ arayüzü varsa, hangisinden gönderileceğini belirler. Ağ arayüz paremetresinin kullanımı zorunludur. Örnek bir kullanımı gösterildiği gibi olmaktadır.

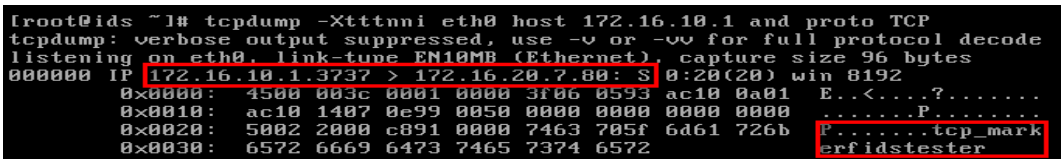
```
# fidstester -i vmnet3
```

[-m] tekil_cümlecik: Gönderilen her paketin içerik (payload) kısmına her protokole özgü cümlecik ekleyerek hedef sisteme gönderilmesi için kullanılır. Bu şekilde ”*fidsd*” uygulaması ”*fidstester*” tarafından gönderilen paketleri ayırt ederek sadece ”*fidstester*” tarafından gönderilen paketlerin işlenmesine olanak sağlayacaktır. Ön tanımlı olarak kullanılan cümleciklere -m paremetresi ile eklenen cümlecik, gönderilecek olan paketin içerik (payload) kısmına eklenir. Eklenebilecek olan cümlecikler Tablo 4.1’de gösterilmektedir.

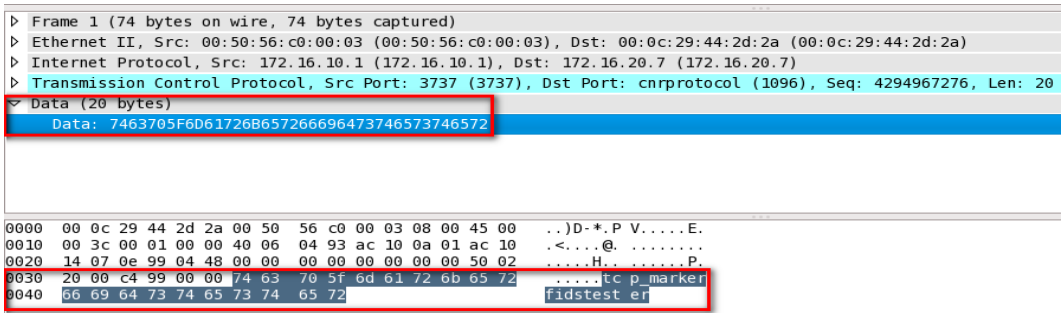
Tablo 4.1. Payload Kısmına Eklenen Cümlecikler

Protokol	**Kullanılan Tekil Cümlecik**
Tcp	tcp_marker
Ucp	udp_marker
Icmp	icmp_marker

Paket gönderimi esnasında ”*-m*” paremetresi ile eklenen tekil cümlecik paketin içerik (payload) kısmında görüntülenebilmektedir. Bu duruma ilişkin örnek bir çıktı hem ”*tcpdump*” yazılımı ile hem de ”*wireshark*” yazılımı ile ele alınmıştır.



Şekil 4.2. Tekil Cümlecik Kullanımı ve Tcpdump Çıktısı



Şekil 4.3. Tekil Cümlecik Kullanımı ve Wireshark Çıktısı

Şekil 4.2 ve Şekil 4.3’de görüldüğü gibi kullanılan tekil cümlecik paketin veri kısmında açıkça görülmektedir. Aynı şekilde ”*fidsd*” uygulamasının komut satırı paremetrelerinden biri olan “*-m*” ile aynı tekil cümlecik ile kullanılarak sadece “*fidstester*” tarafından gönderilen paketleri işlenmesi sağlanmış olacaktır. Tekil cümlecik kullanımı zorunludur. Örnek bir kullanım gösterildiği gibi olmaktadır.

```
# fidstester -m fidstester
```

[-d] gecikme_zamanı: Özellikle durum korumalı tehdit gözetleme sistemlerinin test adımlarının gerçekleştirildiği, yapılandırma dosyası içerisinde ”*ids-conn*” ile belirtilen komut satırı kullanımında ”*-d*” paremetresi ile kullanımının gerçekleştirildiği gecikme zamanının saniye cinsinden belirtildiği paremetredir. Örnek bir kullanımı gösterildiği gibi olmaktadır.

```
# fidstester -d 1.0
```

”*Fidstester*” uygulamasına ait paremetrelerin tümünün kullanımına ait örnek bir kullanım topluca aşağıda gösterildiği gibi olmaktadır.

```
# fidstester -f fidstester.conf -i vmnet3 -m fidstester -d 1.0
```

Özellikle durum korumalı tehdit gözetleme sistemlerinin test adımlarında kullanılan ağ topolojisinin durumuna bağlı olarak uygun bir değer kullanımı gereklidir.

4.2.2. Fidstester yapılandırma dosyası

”*-f*” parametresi ile belirtilen dosya ”*fidstester*” uygulamasının paketleri oluşturup göndermek için kullanacağı yapılandırma dosyası olacaktır. Bu dosyanın formatı ”***:***” ile birbirinden ayrılmış Tcp/Ip alanlarıdır. Her bir test edilecek sistemlere özgü olarak uygun değerler ile doldurulmalıdır. 172.16.20.5 hedef ip adres bilgisine sahip sistemin 80/Tcp portuna, 172.16.10.1 kaynak ip adres bilgisi ve 3737 kaynak port numarası ile paket göndermek için, yapılandırma dosyası içerisinde aşağıda gösterildiği biçimde belirtimi gerçekleştirilmelidir.

```
172.16.10.1:3737:172.16.20.5:80:S:TCP
```

Yapılandırma dosyasına özgü değerler ve kullanımları alt başlıklar halinde anlatılacaktır.

Açıklama Satırı: "#" ile yada ";" ile başlayan satırlar fidstester yazılımı tarafından dikkate alınmaz. Bu karakterler ile başlayan satırlar açıklama satırı olarak ele alınmaktadırlar. Örnek bir kullanım aşağıda gösterilmiştir. "#" karakteri ile başlayan TCP Portlarının Taranması satırı "*fidstester*" uygulaması tarafından açıklama satırı olarak ele alınmaktadır.

```
# TCP Portlarının Taranması
#172.16.10.1:3737:172.16.20.5:23-25:S:TCP
```

Stop_Signal: Fidstester" uygulamasının yapılandırma dosyasında belirtilen değerlere uygun paketlerin gönderilmesinin ardından, "*fidsd*" uygulamasının paketleri işleyip durması için gerekli özel olarak oluşturulmuş "*icmp*" paketidir. Bu paketi alan "*fidsd*" yazılımı "*fidstester*" uygulamasının yapılandırma dosyasındaki paketlerin gönderim işleminin tamamlandığını anlamakta ve çalışmasını sonlandırmaktadır.

"*stop_signal*" için sadece "*icmp*" paketi kullanılmaktadır. Bunun için "*fidsd*" uygulamasının çalıştığı sisteme belirtilen "*icmp*" paketlerinin geçişine izin verilmelidir. "*pf*" için kullanılabilecek güvenlik duvarı kuralı gösterildiği gibi olmaktadır.

```
pass in on $ext_if proto icmp from 172.16.10.1 to 172.16.20.0/24
```

"*stop_signal*" paketinin veri kısmına "*fidstester-py*" eklenerek hedef sisteme gönderilmektedir. Bu işlemin gerçekleştiği kod bloğu gösterildiği gibi olmaktadır.

```
stop_signal_string="fidstester-py"
```

```
...
if re.match(self.reg_stop_signal,line):
                main_packet=line.split("=")[1]
                pkt=self.createPackets(main_packet,"ICMP")

packet=IP(src=pkt[0],dst=pkt[1])/ICMP(type=int(pkt[3]),code=int(pkt[4]))/
Raw(stop_signal_string)
        send(packet)
```

”*stop_signal*” özel paketi ”*fidsd*” uygulamasının çalışmasını durduracağı için *”-f*” paremetresi ile belirtilen yapılandırma dosyasının en alt satırında kullanılmalıdır. Örnek bir kullanım gösterildiği gibi olmaktadır.

```
stop_signal=172.16.10.1::172.16.20.4:::ICMP:8:0
```

Güvenlik Duvarı Kuralları: ”*ids*” ve ”*ids-conn*” ile başlayan satırlar ”*ids*” test adımlarını içeren değerlerdir. Aynı şekilde ”*#*” ve ”*;*” karakteri ile başlayan satırlarda ”*fidstester*” uygulaması tarafından açıklama satırı olarak ele alınmaktadır. Bunların dışında kalan satırlar ise güvenlik duvarı test adımlarını içeren satırlar olmaktadır. Bunun için özel bir karakter kullanılmamaktadır.

Her satır ”*:*” Karakteri ile birbirinden ayrılmış Tcp/Ip alanlarıdır. Ağ topolojisine uygun olarak belirlenecek olan değerler ile oluşturulmalıdır. Genel formatı aşağıda gösterildiği gibi olmaktadır.

”*Tcp*” ve ”*udp*” prorokolleri için gösterildiği gibi olmaktadır. ”*Udp*” protokolünde bayrak kullanımı yer almadığı için ilgili alan boş bırakılmalıdır.

Kaynak_Ip:Kaynak_Port: Hedef_Ip:Hedef_Port/Aralığı:Bayrak:Protokol(Tcp/Udp)

Port bölümünde yer alan ilgili bölüm için port aralığı da kullanılabilmektedir. Yani 1, 1024 arası portların kullanımı için port bölümü 1-1024 olarak kullanılmalıdır.

172.16.20.5 hedef ip adresinin 1, 1024 arası portlarının ”*syn*” bayrağının aktif olarak taranması için yapılandırma dosyasında yer alması gereken örnek satır gösterildiği gibi olmalıdır.

```
172.16.10.1:3737:172.16.20.5:1-1024:S:TCP
```

Aynı hedef sistemin 1,1024 arası port numarasına sahip ”*udp*” portlarının taranması için de yapılandırma dosyası içerisinde yer alması gereken örnek satır gösterildiği gibi olmaktadır.

```
172.16.10.1:3737:172.16.20.5:1-1024::UDP
```

Görüldüğü gibi bayrak kısımı ”*tcp*” protokolüne özgü bir kullanım olduğu için ilgili alan boş olarak bırakıldı ve son kısım olan protokol kısmında ise ”*tcp*” olan belirtim ”*udp*” olarak gerçekleştirilmiştir. ”*Tcp*” protokolüne özgü bayrak kullanımının, ”*fidstester*” uygulamasındaki kullanımı için Tablo 4.2’de yer alan değerler kullanılmalıdır.

Tablo 4.2. Tcp Protokolüne Ait Bayrak Değerlerinin Fidstester İçin Kullanımı

Bayrak Adı	**Yapılandırma Dosyasındaki Değeri**
Syn	S
Ack	A
Fin	F
Rst	R
Urgent	U
Push	P

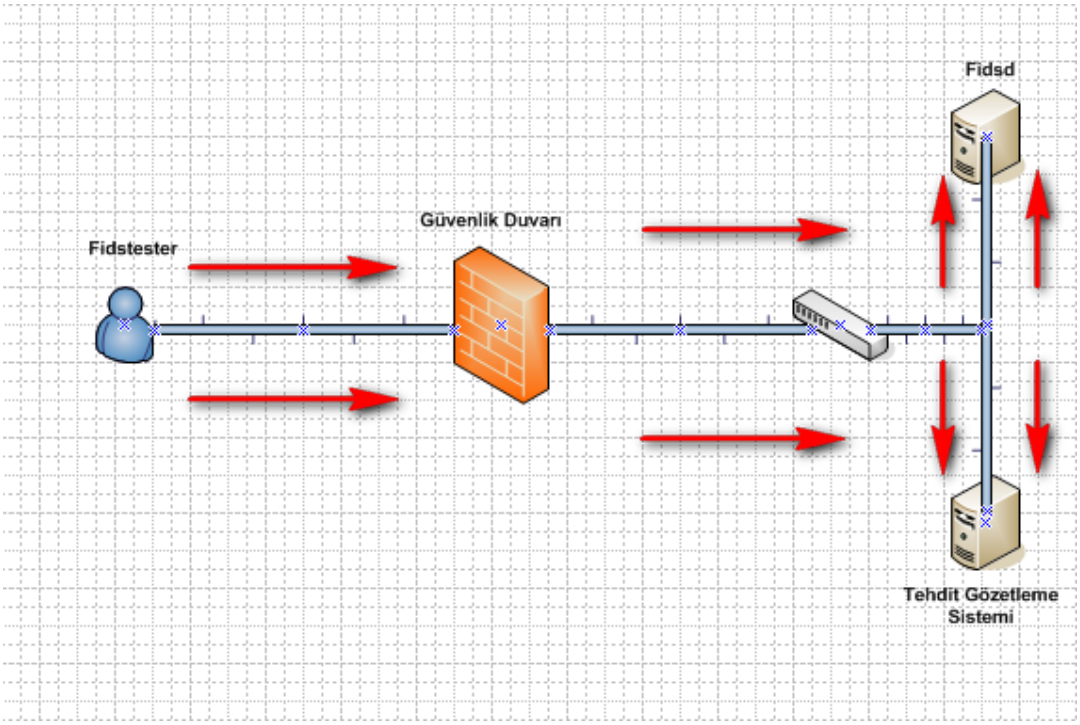
Hedef port değeri için port aralığı kullanılabildiği gibi tek bir port numarası da kullanılabilmektedir. Bu kullanıma dair örnek kurallar için yapılandırma dosyası içerisinde yer alması gereken satırlar gösterildiği gibi olmaktadır.

```
172.16.10.1:3737:172.16.20.9:21:S:TCP
172.16.10.1:3737:172.16.20.5:53:UDP
```

İlk kuralda 172.16.20.9 hedef ip adres bilgisine sahip sistemin 21/tcp numaralı portuna syn bayrağı aktif hale getirilmiş ilgili paket gönderilmektedir. İkinci kural tanımında ise 172.16.20.5 hedef ip adres bilgisine sahip sistemin 21/udp numaralı portuna ilgili paket gönderilmektedir. Her iki kuralda da kaynak ip adres bilgisi olarak 172.16.10.1 ve kaynak port numarası belirtimi olarak da 3737 belirtilmiştir.

Tehdit Gözetleme Sistemi Kuralları: Yapılandırma dosyası içerisinde "*ids*" ve "*ids-conn*" ile başlayan satırlar tehdit gözetleme sistemlerinin test adımlarını içeren kurallar için kullanılmaktadır. Tehdit gözetleme sistemi için sadece "*-f*" paremetresi ile belirtilen yapılandırma dosyası içerisinde ilgili ağ topolojisine uygun kurallar oluşturularak "*fidstester*" uygulaması çalıştırılmalıdır. Tehdit gözetleme sistemi için "*fidsd*" uygulamasının çalıştırılmasına gerek bulunmamaktadır. Çünkü hem "*fidstester*" uygulaması hemde "*fidsd*" uygulaması tehdit gözetleme sistemlerinin test adımlarını içeren kurallar ile oluşturulan paketlerin kayıt işlemlerini tutmamaktadırlar.

Tehdit gözetleme sisteminin ağ topolojisi içerisindeki konumu hedef sistemlere giden ve gelen bütün paketleri işleyecek şekilde konuşlandırılmalıdır. Kullanılan teknik cihazların yeteneklerine göre bu işlem gerçekleştirilmelidir. Burada oluşturulan sanal labratuar ortamında kullanılan sanallaştırma teknolojisi yazılımı olan Vmware Workstation ile bu işlem aynı sanal ağ arayüzlerinin kullanılması ile sağlanmıştır. Kullanılan ağ topolojisi için tehdit gözetleme sisteminin konumu ve hedef sisteme gelen ve giden ağ trafiğinin akışı Şekil 4.4'de gösterilmektedir.



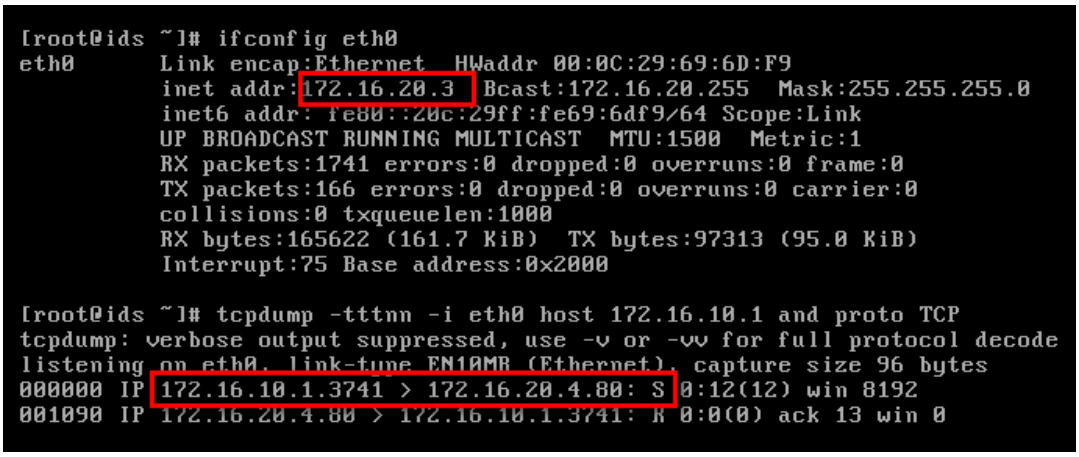
Şekil 4.4. Tehdit Gözetleme Sisteminin Ağ Topolojisi İçerisindeki Konumu

Tehdit gözetleme sistemi test adımları için kullanılabilecek "*ids*" ve "*ids-conn*" olmak üzere iki seçenek bulunmaktadır. Bunlardan birincisi olan ids ile tehdit gözetleme sisteminin durum koruması özelliğine sahip olup olmadığı test edilmektedir. İkinci seçenek olan "*ids-conn*" ile de 3 yollu el sıkışmanın tamamlanmasının ardından ilgili paket hedef sisteme gönderilmektedir. Bu şekilde durum koruması özelliği sağlanarak paketin tehdit gözetleme sistemi tarafından işlenmesine olanak sağlanmış olacaktır.

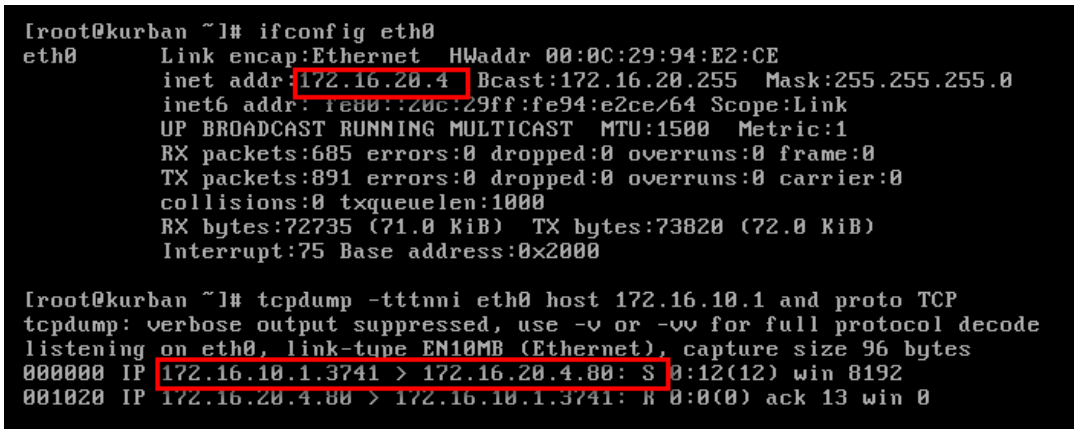
Tehdit gözetleme sisteminin durum koruma özelliğinin test edilmesi için kullanılan "*ids*" seçeneği ile oluşturulan paketlerin kural tanımlaması güvenlik duvarı kural tanımlaması ile aynıdır. Sadece kuralın başında "*ids*=" tanımlamasının yapılması yeterlidir. Bu duruma örnek teşkil edecek bir kural tanımı gösterildiği gibi olmaktadır.

```
ids=172.16.10.1:3471:172.16.20.4:80:S:TCP:/etc/passwd
```

Bu kural tanımı ile 172.16.20.4 hedef ip adres bilgisine sahip sistemin 80/tcp portuna 172.16.10.1 kaynak ip adres bilgisi ve 3471 kaynak port bilgisi ile veri kısmında ”*/etc/passwd*” içeriği bulunan paket gönderilmektedir. Bu paketi işleyen tehdit gözetleme sistemi durum koruma özelliği aktif ise 3 yollu el sıkışma tamamlanmadan paket gönderildiği için paketi işlenmeyecektir. İlgili paketin gönderilmesinin ardından 172.16.20.4 ip adres bilgisine sahip sistem ve tehdit gözetleme sisteminin bulunduğu 172.16.20.3 ip adres bilgisine sahip sisteminin üzerindeki tcpdump çıktıları Şekil 4.5 ve Şekil 4.6'da görülmektedir.

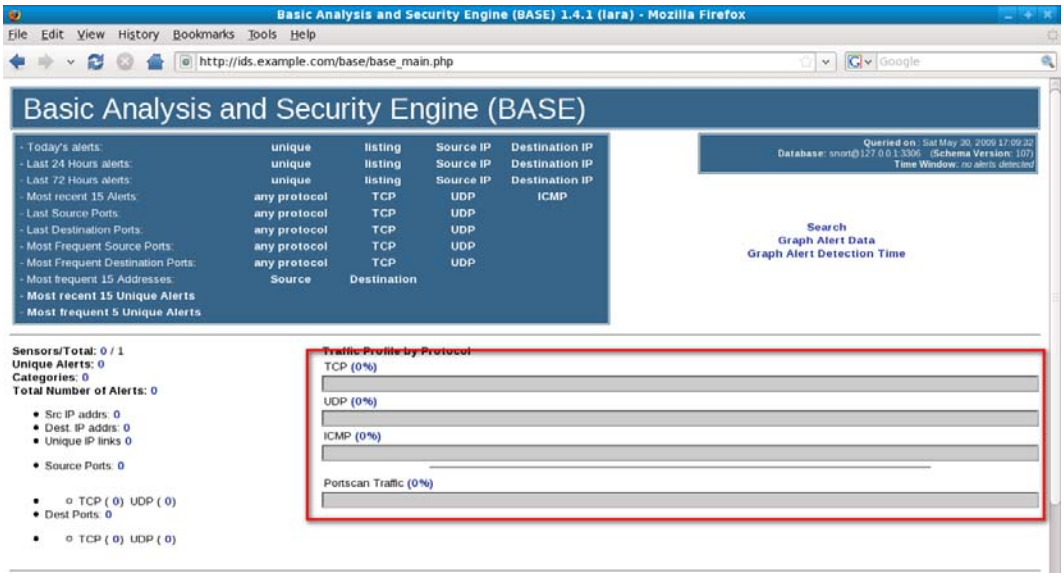


Şekil 4.5. Tehdit Gözetleme Sistemi Durum Koruma Özelliği Adımı ve 172.16.20.3 Sistemi Üzerinde Tcpdump Çıktısı



Şekil 4.6. Tehdit Gözetleme Sistemi Durum Koruma Özelliği Adımı ve 172.16.20.4 Sistemi Üzerinde Tcpdump Çıktısı

Son olarak ilgili kural ile oluşturulan paketin modellenen ağ topolojisi üzerinde tehdit gözetleme sistemi olarak kullanılan ”snort” servisi tarafından hiç bir uyarı üretmediği de snort servis kayıt işlemlerini web arabiriminden izlemek için kullanılan base uygulaması ile görüntülenerek doğrulanmıştır. Bu durum Şekil 4.7’de görülmektedir.

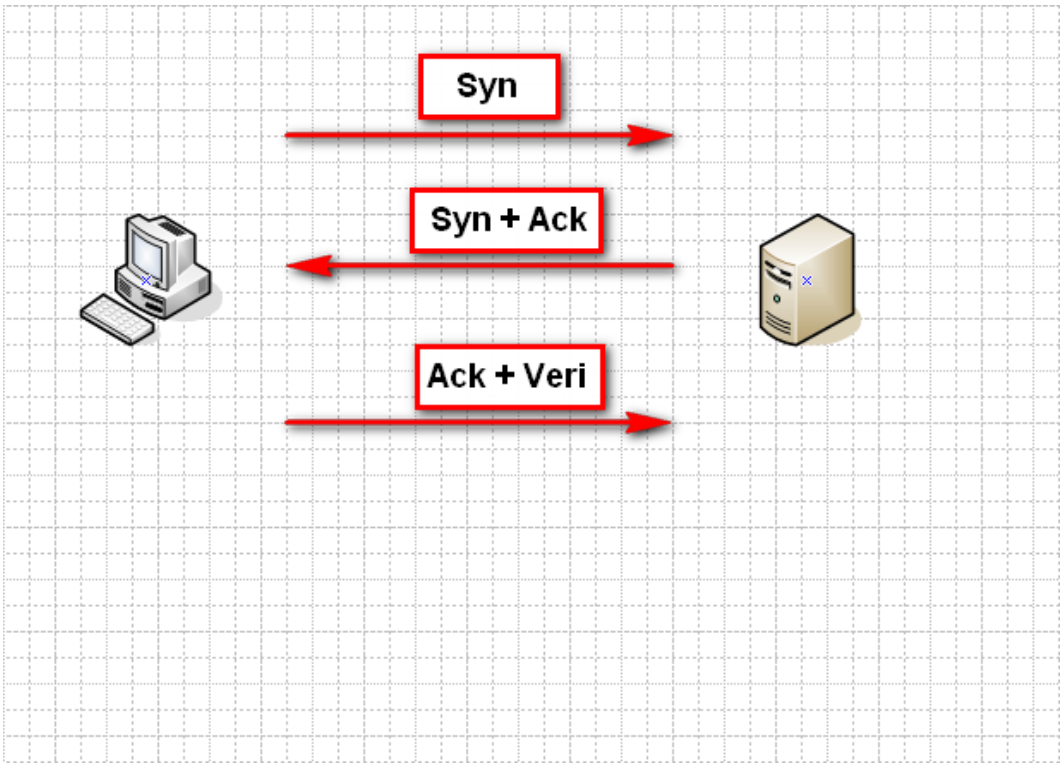


Şekil 4.7. Tehdit Gözetleme Sistemi Durum Koruma Özelliği Adımı ve Snort Kayıt İşlem Bilgileri

Yine güvenlik duvarı kurallarında olduğu gibi hedef sistem için port aralığı kullanılabilmektedir. Bayrak bölümü için Tablo 4.2’de bulunan değerler geçerli olmaktadır.

Durum koruma özelliği bulunan tehdit gözetleme sistemlerinin test edilmesi için kullanılan “*ids-conn*” seçeneği ile oluşturulan paketlerin kural tanımlaması güvenlik duvarı kural tanımlaması ile aynıdır. Sadece kuralın başında “*ids-conn=*” tanımlamasının yapılması yeterlidir. Bu duruma örnek teşkil edecek bir kural tanımı gösterildiği gibi olmaktadır. “*ids-conn*” seçeneği ile oluşturulan paketlerin diğer kurallara göre oluşturulan paketlerden farkı hedef sistem ile 3 yollu el sıkışmanın tamamlansı ile verinin gönderilmesidir.

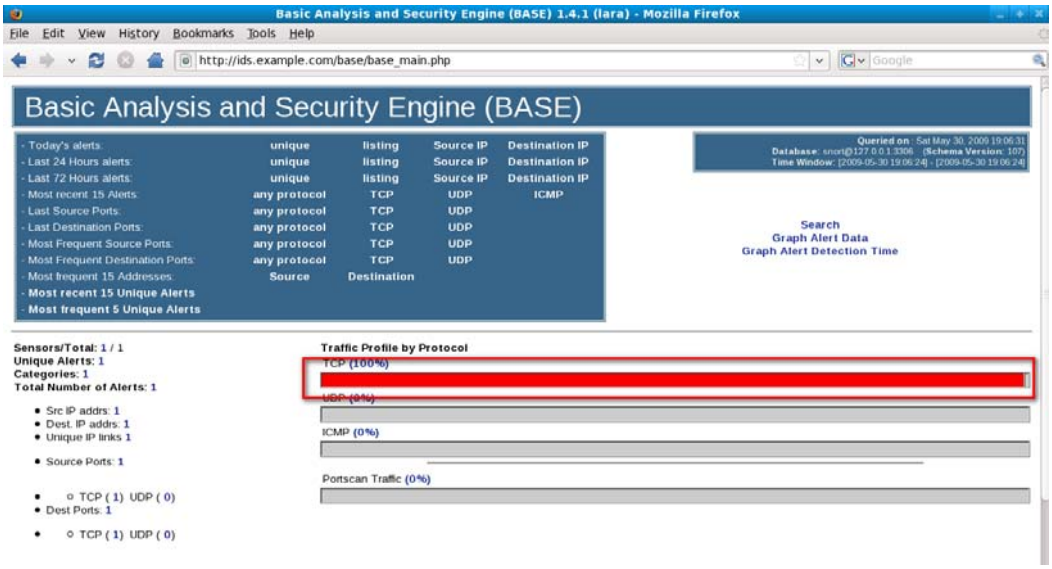
3 yollu el sıkışma "*tcp*" protokolüne özgü bir davranıştır. "*Udp*" ya da "*icmp*" protokolünde durum koruması kavramı dolayısı ile 3 yollu el sıkışma yoktur. Kısaca 3 yollu el sıkışma hedef sisteme "*syn*" bayrağı aktif hale getirilmiş paket gönderilerek başlar. Ardından hedef sistem "*ack*" bayrağı aktif hale getirilmiş paket göndererek cevap verir. Son olarak yine hedef sisteme "*ack*" bayrağı aktif hale getirilmiş ve veri kısmında istenen veri ile birlikte paket gönderilerek 3 yollu el sıkışma tamamlanarak paket gönderilir. Bu durum Şekil 4.8'de gösterilmiştir. "*Fidstester*" için dikkat edilmesi gereken bir husus ise "*-d*" paremetresi ile belirlenen gecikme zamanı değerinin kullanılan ağ topolojisine uygun bir biçimde seçilmesidir.



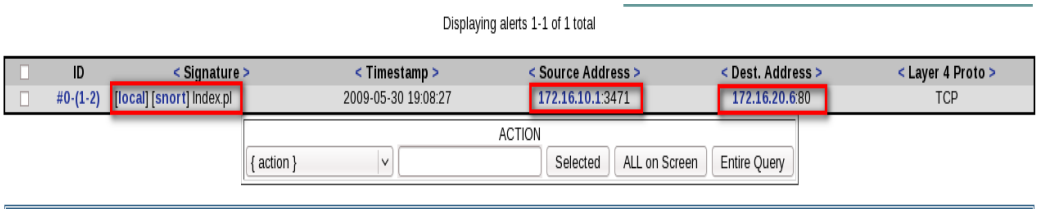
Şekil 4.8. 3 Yollu El Sıkışma

```
ids-conn=172.16.10.1:3471:172.16.20.6:80:S:TCP:index.pl
```

Bu kural tanımı ile 172.16.20.6 hedef ip adres bilgisine sahip sistemin 80/tcp portuna 172.16.10.1 kaynak ip adres bilgisi ve 3471 kaynak port bilgisi ile veri kısmında ”*index.pl*” içeriği bulunan paket 3 yollu el sıkışmanın tamamlanmasının ardından gönderilmektedir. İlgili paketin gönderilmesinin ardından ilgili paketin tehdit gözetleme sistemi log kayıt bilgisi Şekil 4.9 ve Şekil 4.10’da görüldüğü gibi olmaktadır.



Şekil 4.9. 3 Yollu El Sıkışma İle Paketlerin Gönderilmesi ve Snort Kayıt İşlem Bilgileri - 1



Şekil 4.10. 3 Yollu El Sıkışma İle Paketlerin Gönderilmesi ve Snort Log Kayıt İşlem Bilgileri - 2

4.2.3. Fidstester kayıt işlem dosyası

”*Fidstester*” yapılandırma dosyasında belirlenen kurallara göre ilgili paketleri oluşturarak hedef sistemlere gönderir. Paket gönderme işlemi ister başarılı olsun isterse başarısız olsun gönderilen her paket için işlem kaydı oluşturur. Yalnız bu işlem sadece güvenlik duvarı kuralları için geçerlidir. Tehdit gözetleme sistemi kuralları için işlem kaydı oluşturulmaz. Kayıt işlemleri ”*fidstester.log*” dosyasında oluşturulur. Bu durum Şekil 4.11’de görülmektedir.

Kayıt işlemleri için belirlenen format ;

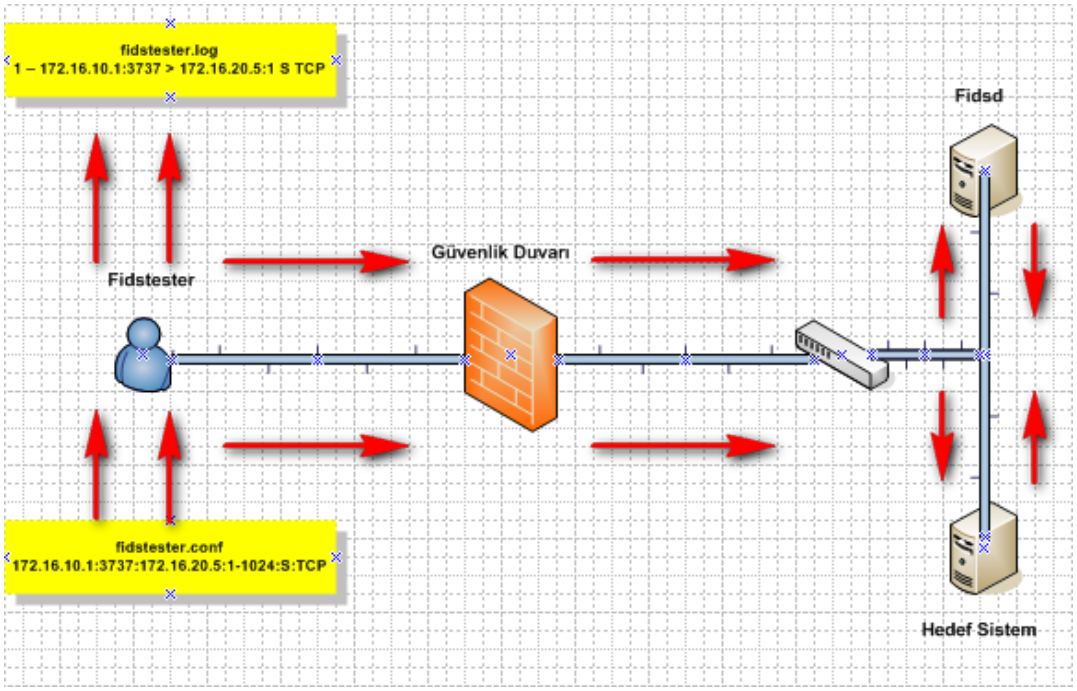
Tekil_Numara - Kaynak_Ip:Kaynak_Port > Hedef_Ip:Hedef_Port Bayrak Protokol

Gönderilen her paket için tekil bir numara kullanılmıştır. “*Fidsd*” uygulaması tarafından da aynı format kullanılmıştır. Bu şekilde “*fidsreport*” uygulaması kayıt işlemleri için tutulan dosyaları karşılaştırarak güvenlik duvarı kural tablosunu görüntülemektedir. Aşağıda gösterilen kural tanımları için fidstester uygulamasının çalıştırılmasının ardından oluşan kayıt dosyası görüldüğü gibi olmaktadır.

```
172.16.10.1:3737:172.16.20.7:443:S:TCP
172.16.10.1:3737:172.16.20.9:21:S:TCP
172.16.10.1:3737:172.16.20.8:53::UDP
```

172.16.20.7 hedef sisteminin 443/tcp portuna, 172.16.20.9 hedef sisteminin 21/tcp portuna ve son olarak 172.16.20.8 hedef sisteminin 53/udp portuna ilgili paketler gönderilmiş ve ”*fidstester*” tarafından oluşturulan kayıt bilgileri aşağıda görüldüğü gibi olmaktadır.

```
1 - 172.16.10.1:3737 > 172.16.20.7:443 S TCP
2 - 172.16.10.1:3737 > 172.16.20.9:21 S TCP
3 - 172.16.10.1:3737 > 172.16.20.8:53 UDP
```



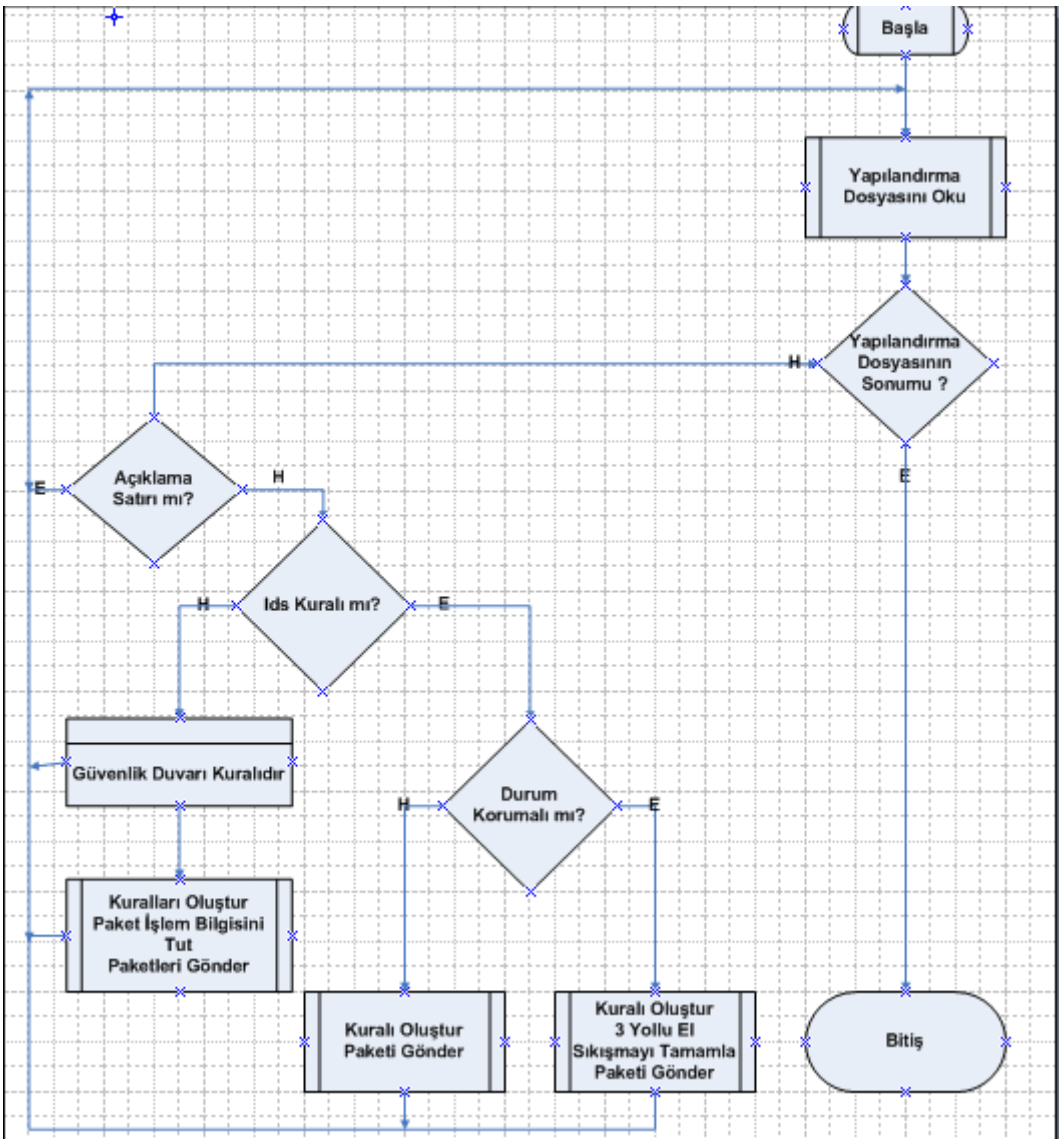
Şekil 4.11. Fidstester İşlem Kayıtlarının Tutulması

"*Fidstester*" uygulaması tarafından oluşturulan paketlerin işlem kayıtlarının tutulmasına ilişkin kod bloğu aşağıda görüldüğü gibi olmaktadır.

```
def writeLog (self,packet,proto):
        """ Write packets into the log file """

        log_file=open(self.ftest_log_file,"a")

        if proto == "TCP":
            log_file.write(str(packet[0]) + " - " + packet[1] + ":" + packet[2] + "
> " + packet[3] + ":" + str(packet[4]) + " " + packet[5] + " " + proto + "\n")
            log_file.close()

        elif proto == "UDP":
            log_file.write(str(packet[0]) + " - " + packet[1] + ":" + packet[3] + "
> " + packet[2] + ":" + str(packet[4]) + " " + proto + "\n")
            log_file.close()

        elif proto == "ICMP":
```

```
            log_file.write(str(packet[0]) + " - " + packet[1] + " > " + packet[2] +
" " + proto + " " + packet[3] + " " + packet[4])
            log_file.close()
```

”*Fisdtester*” çalışmasının akış şeması Şekil 4.12’de gösterildiği gibi olmaktadır.

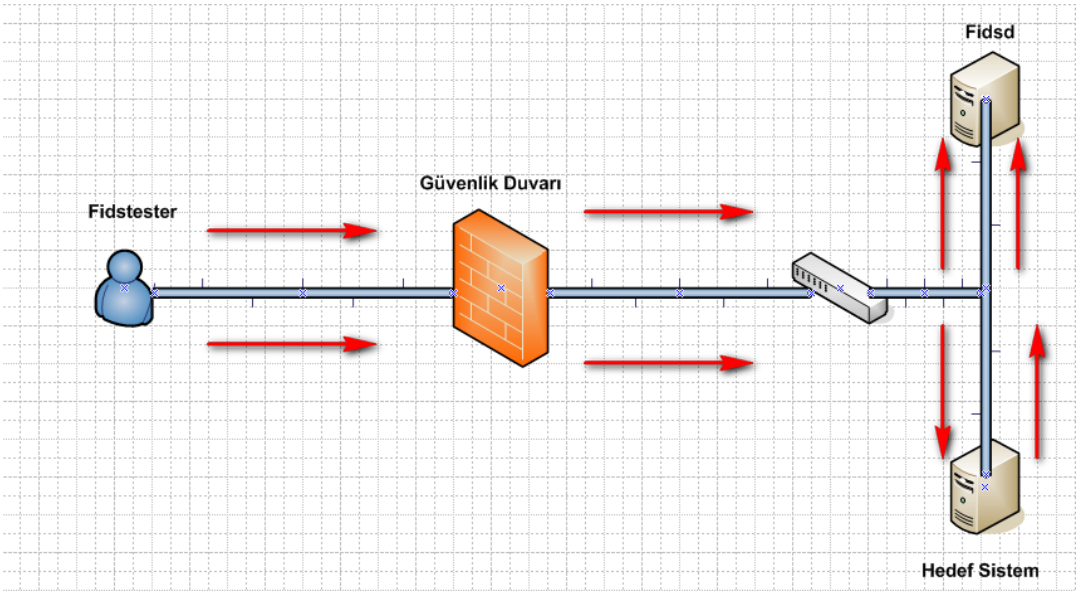


Şekil 4.12. Fidstester Akış Şeması

4.3. Fidsd

”*Fidsd*” uygulaması özel olarak oluşturulmuş ”*stop_signal*” paketini işleyene kadar, ”*fidstester*” uygulamasının belirtilen yapılandırma dosyasına göre oluşturarak hedef sisteme gönderdiği paketleri işleyerek kayıt altına alan uygulamadır. Kayıt işlemleri için kullandığı dosya “*fidsd.log*” olmaktadır.

”*Fidsd*” uygulaması tehdit gözetleme sisteminin de olduğu gibi hedef sistemlere giden gelen bütün trafiği dinleyebilecek bir yapıda olmalıdır. Bu işlem kullanılan teknik cihazlara göre gerçekleştirilmelidir. Bu çalışma da kullanılan ”*fidsd*” uygulamasının ağ topolojisindeki konumu Şekil 4.13’de görüldüğü gibi olmaktadır.



Şekil 4.13. Fidsd Uygulamasının Ağ Topolojisindeki Konumu

4.3.1. Fidsd kullanım paremetreleri

”*Fidstester*” uygulamasına ait kullanım paremetreleri ve bu paremetrelere ait açıklamalar gösterildiği gibi olmaktadır.

[-i] ağ_arayüzü: ”*Fidsd*” uygulamasının hangi ağ arayüzünü dinleyerek çalışacağının belirlendiği paremetredir. Eğer birden fazla ağ arayüzüne sahip bir sistemde ”*fidsd*” uygulaması çalıştırılıyor ise ağ trafiğinin dinleneceği ağ arayüzü ”-*i*” paremetresi ile belirtilmelidir. Ön tanımlı değeri ”*eth0*”’dır. Eğer birden fazla ağ arayüzü varsa ve bu arayüz ”*eth0*” olmayacak ise belirtimi zorunludur. Örnek bir kullanım için;

```
# fidsd -i eth0
```

[-m] tekil_cümlecik: Fidstester uygulaması tarafından ”-*m*” paremetresi ile özel veri alanı ile oluşturulan paketlerin, ”*fidsd*” uygulaması tarafından işlenmesi için gerekli paremetredir. Bu şekilde sadece ”*fidstester*” uygulaması tarafından gönderilen paketlerin ”*fidsd*” tarafından işlenmesi sağlanmaktadır. Burada belirtilen tekil cümlecik ”*fidstester*” uygulamasında kullanılan tekil cümlecik ile aynı olmak zorundadır. Örnek bir kullanım gösterildiği gibi olmaktadır.

```
# fidsd -m
```

”*Fidsd*” uygulamasına ait paremetrelerin tümünün kullanımına ait örnek bir kullanım topluca aşağıda gösterildiği gibi olmaktadır.

```
# fidsd -i eth0 -m fidstester
```

”*Fidsd*” uygulamasının ”*fidstester*” uygulaması tarafından özel veri alanı ile gönderilen paketleri nasıl işlediği için kullanılan kod bloğu aşağıda görüldüğü gibi olmaktadır.

```
...
tcp_marker="tcp_marker" + self.options.marker
udp_marker="udp_marker" + self.options.marker
icmp_marker="icmp_marker" + self.options.marker
...
tcp_reg = re.compile(".*%s"% tcp_marker)
udp_reg = re.compile(".*%s"% udp_marker)
icmp_reg = re.compile(".*%s"% icmp_marker)
if proto == "tcp" and re.match(tcp_reg,data):
...
elif proto == "udp" and re.match(udp_reg,data):
...
elif proto == "icmp" and re.match(icmp_reg,data):
...
```

4.3.2. Fidsd kayıt işlem dosyası

”*Fidstester*”, yapılandırma dosyasında belirlenen kurallara göre ilgili paketleri oluşturarak hedef sistemlere gönderir. Bu paketlerin veri kısmı ”*-m*” paremetresi ile belirlenen tekil bir cümlecik ile doldurularak hedef sistemlere gönderilir. ”*Fidsd*” uygulaması ağ topolojisi üzerindeki konuşlandırmasına ve ”*-i*” paremetresi ile belirlenen ağ arayüzüne göre gelen bu ağ trafiğini yine veri kısmında özel bir cümlecik geçen ”*stop_signal*” paketini işleyene kadar dinlmeye başlar. Bu sırada işlediği her paket için işlem kaydı tutar. Kayıt işlemleri ”*fidsd.log*” dosyasında oluşturulur. Bu durum Şekil 4.14’ de görülmektedir.

Kayıt işlemleri için belirlenen format ”*fidstester*” uygulaması tarafından oluşturulan işlem kayıt dosyası formatı ile aynıdır.

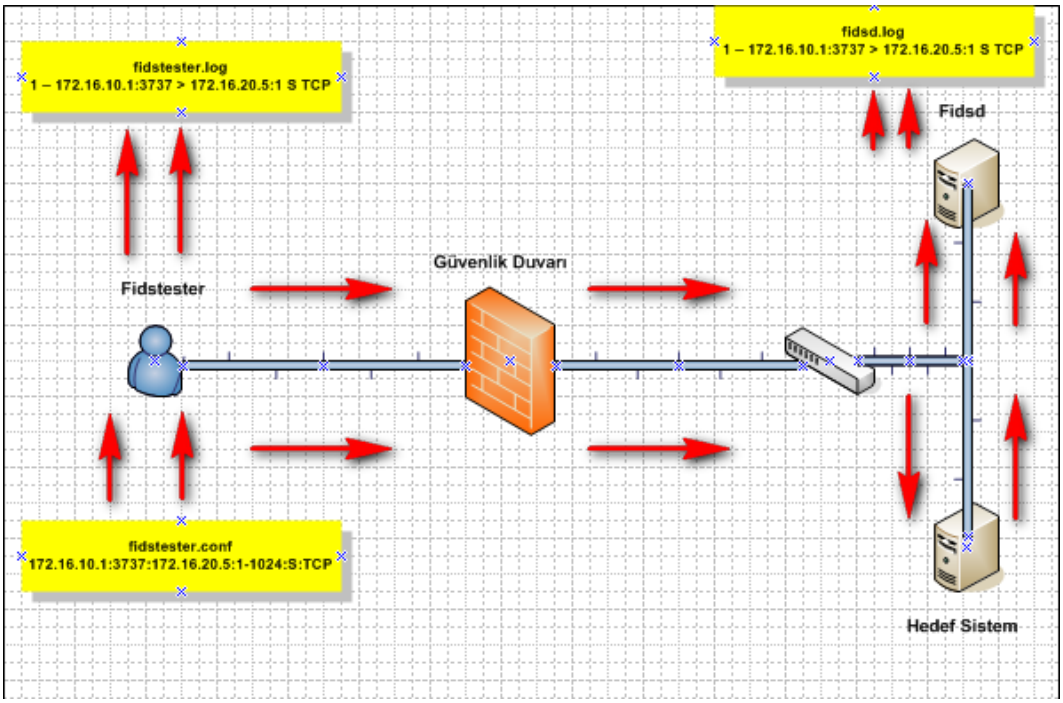
Tekil_Numara - Kaynak_Ip:Kaynak_Port > Hedef_Ip:Hedef_Port Bayrak Protokol

Gönderilen her paket için tekil bir numara kullanılmıştır. "*Fidsd*" uygulaması tarafından da aynı format kullanılmıştır. Bu şekilde "*fidsreport*" uygulaması kayıt işlemleri için tutulan dosyaları karşılaştırarak güvenlik duvarı kural tablosunu görüntülemektedir. Aşağıda gösterilen kural tanımları için "*fidstester*" uygulamasının çalıştırılmasının ardından oluşan kayıt dosyası görüldüğü gibi olmaktadır.

```
172.16.10.1:3737:172.16.20.7:443:S:TCP
172.16.10.1:3737:172.16.20.9:21:S:TCP
172.16.10.1:3737:172.16.20.8:53::UDP
```

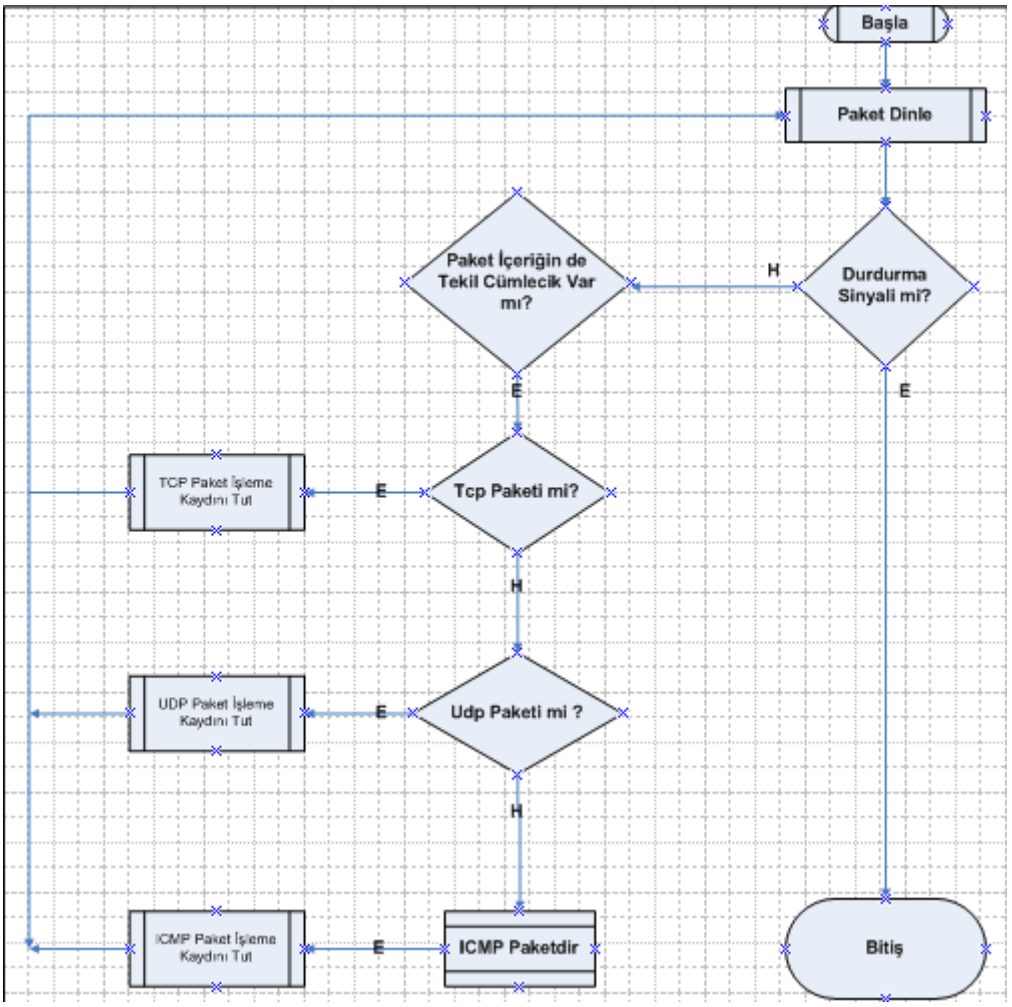
172.16.20.7 hedef sisteminin 443/tcp portuna, 172.16.20.9 hedef sisteminin 21/tcp portuna ve son olarak 172.16.20.8 hedef sisteminin 53/udp portuna ilgili paketler gönderilmiş ve "*fidsd*" uygulaması tarafından oluşturulan kayıt bilgileri aşağıda görüldüğü gibi olmaktadır.

```
1 - 172.16.10.1:3737 > 172.16.20.7:443 S TCP
2 - 172.16.10.1:3737 > 172.16.20.9:21 S TCP
3 - 172.16.10.1:3737 > 172.16.20.8:53 UDP
```



Şekil 4.14. Fidsd İşlem Kayıtlarının Tutulması

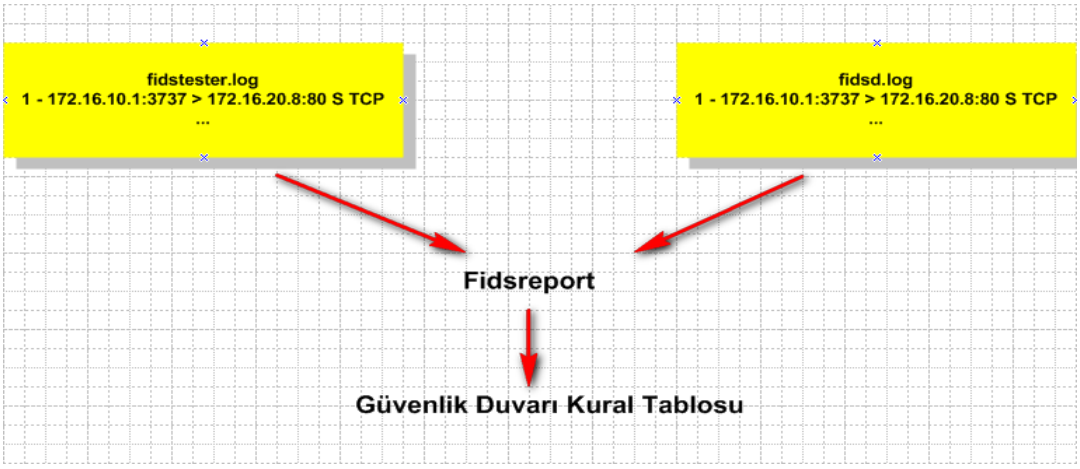
”*Fidsd*” çalışmasının akış şeması Şekil 4.15’de gösterildiği gibi olmaktadır.



Şekil 4.15. Fidsd Akış Şeması

4.4. Fidsreport

”*Fidsreport*” uygulaması “*fidstester*” ve “*fidsd*” uygulamaları tarafından oluşturulan işlem kayıtlarını karşılaştırarak güvenlik duvarı kural tablosunu görüntülemektedir. Bu durum Şekil 4.16’da gösterilmiştir.



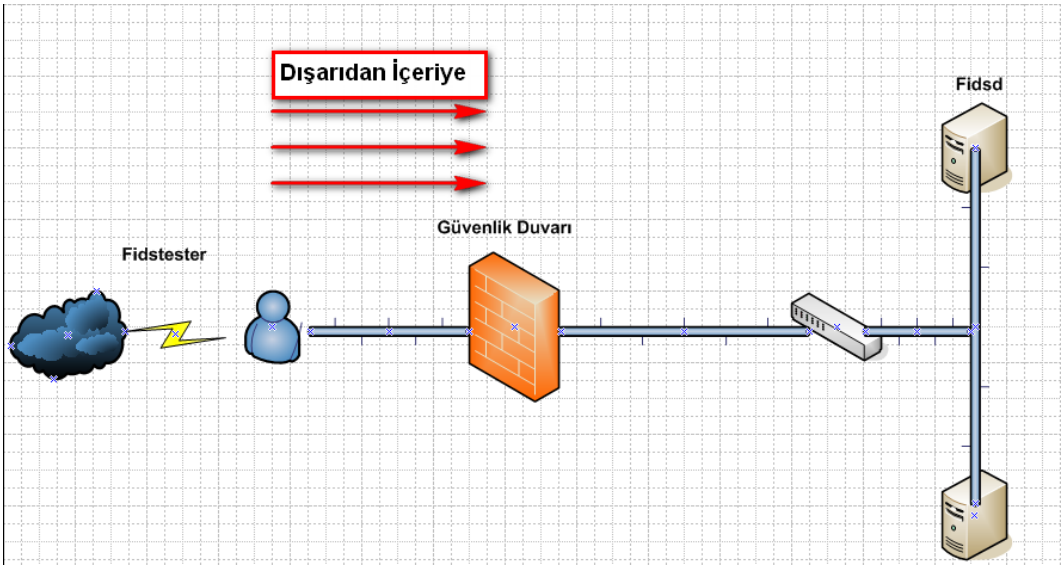
Şekil 4.16. Fidsreport Çalışma Şeması

Kullanımı için iki paremetre kullanmaktadır. Bunlardan birincisi “*fidstester*” uygulaması tarafından oluşturulan işlem kayıt dosyası olan “*fidstester.log*”, diğeri ise fidsd uygulaması tarafından oluşturulan “*fidsd.log*” dosyasıdır. Güvenlik duvarı kural tablosu 3 ana bölümde görüntülenmektedir. Bunlardan birincisi “*fidsreport*” uygulaması tarafından “*Authorized Packets*” olarak listelenen yönlendirme işlemlerinin gerçekleştirildiği güvenlik duvarı kuralları, bir diğeri “*fidsreport*” uygulaması tarafından “*Modified Packets*” olarak listelenen kaynak değiştirme işlemlerinin gerçekleştirildiği güvenlik duvarı kuralları, son olarak ise “*fidsreport*” uygulaması tarafından “*Dropped Packets*” olarak listelenen paket engelleme işlemlerinin gerçekleştirildiği güvenlik duvarı kurallarıdır. Bu durun Tablo 4.3’de listelenmiştir.

Tablo 4.3. Fidsreport Uygulaması Tarafından Yorumlanan Güvenlik Duvarı Kural Tablosu

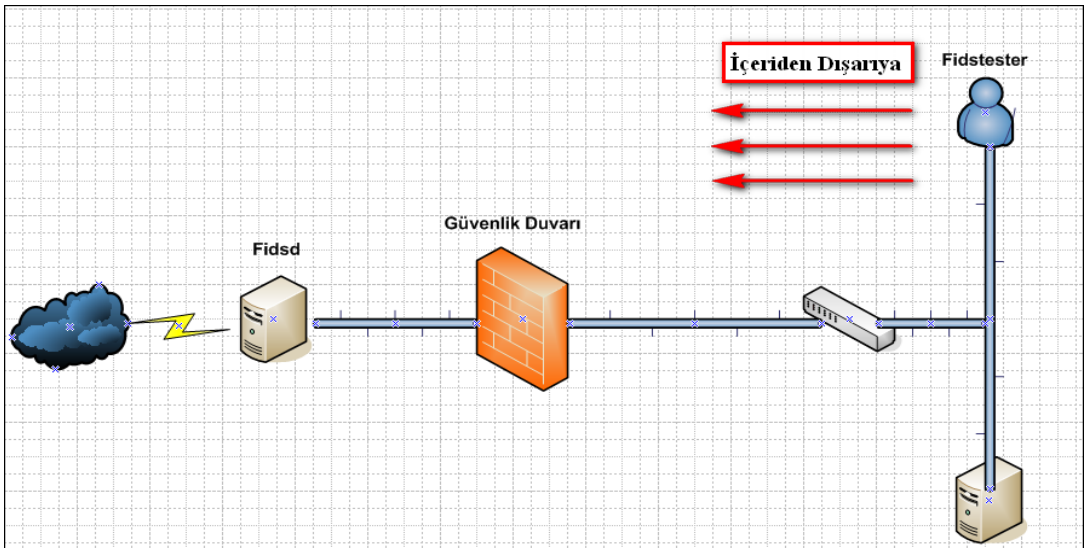
Güvenlik Duvarı Tarafından Paketin Nasıl İşlendiği	**Fidsreport Listeleme Adı**
Yönlendirme İşlemleri	Authorized Packets
Kaynak Değiştirme İşlemleri	Modified Packets
Engelleme İşlemleri	Dropped Packets

"*Fidstester*" ve "*fidsd*" uygulamalarının ağ topolojisindeki konumlarına göre oluşan kayıt işlem dosyalarının içerikleri farklı olacaktır. Bu durum Şekil 4.17 ve Şekil 4.18'de gösterilmiştir.



Şekil 4.17. Fidstester Dışarıdan İçeriye Çalışma Şeması

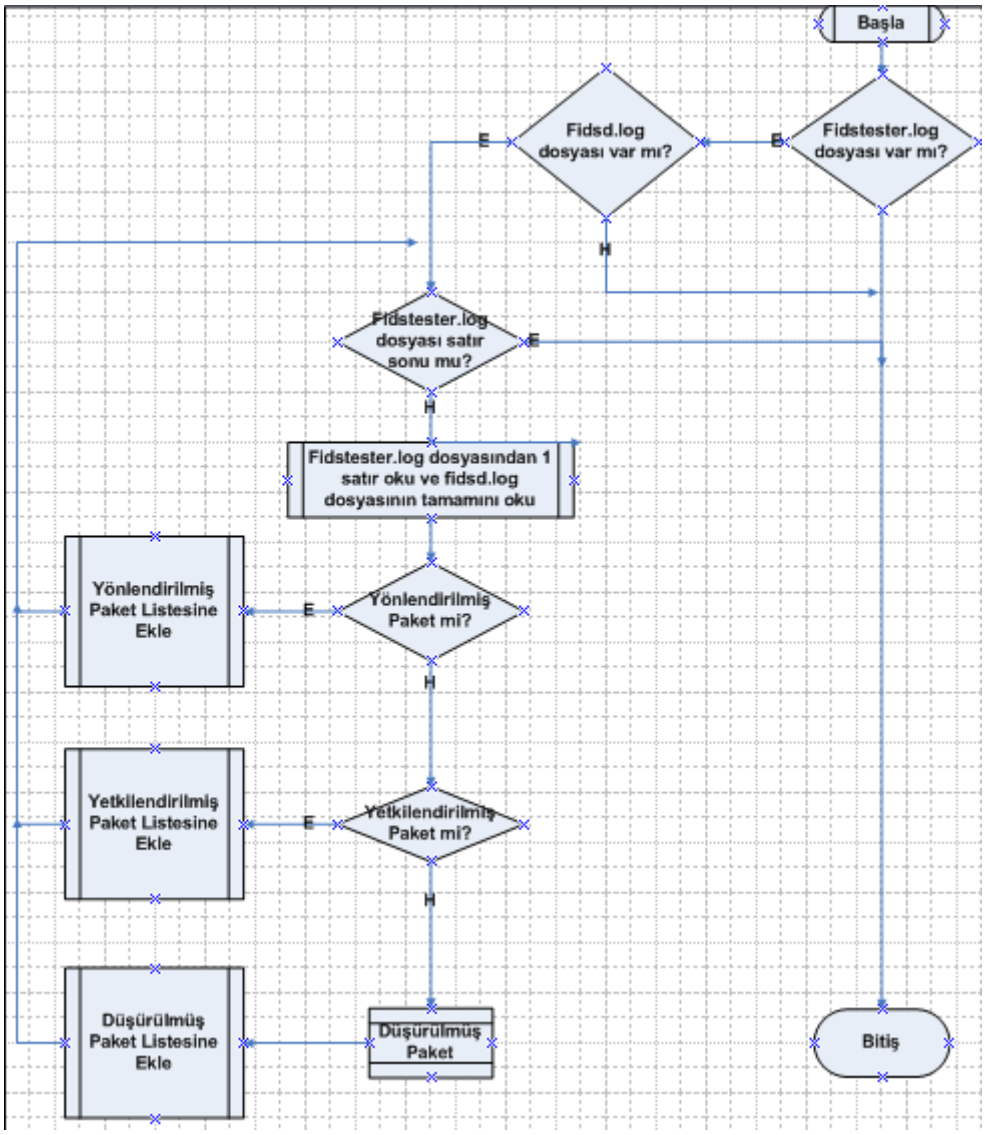
Şekil 4.17'de gerçekleştirilen tarama dışarıdan içeriye doğru tanımlanmaktadır.



Şekil 4.18. Fidstester İçeriden Dışarıya Çalışma Şeması

Şekil 4.18'de gerçekleştirilen tarama içeriden dışarıya doğru tanımlanmaktadır.

"*Fisdreport*" çalışmasının akış şeması Şekil 4.19'da gösterildiği gibi olmaktadır.



Şekil 4.19. Fidsreport Akış Şeması

Her iki durumda da belirtildiği gibi işlem kayıt dosyalarının içerikleri farklı olacaktır. Buna göre de "*fidsreport*" uygulaması tarafından listelenen güvenlik duvarı kural tablosu farklı içeriklere sahip olacaktır. Her iki durum da bu çalışma için kullanılan sanal laboratuar ortamında farklı 2 senaryo ile gerçeklenecektir.

Senaryo - 1:

Aşağıda gösterilen "*pf*" kural tablosuna sahip güvenlik duvarı için Şekil 4.17'de gösterildiği gibi dışarıdan içeriye doğru "*fidstester*" çalıştırılacak ve güvenlik duvarı kural tablosu listelenecektir.

```
ext_if="vic0"
int_if="vic1"
FidsTester="172.16.10.1"
Ftestd="172.16.20.3"
Telnet="172.16.20.5"
Dns="172.16.20.6"
Web="172.16.20.7"
Mail="172.16.20.8"
Ftp="172.16.20.9"
set skip on lo0
block log all
pass out keep state
## External -> Internal

pass in on $ext_if proto icmp from $FidsTester to $Ftestd keep state
pass in on $ext_if inet proto tcp from $FidsTester to $Telnet port 23 keep state
pass in on $ext_if inet proto udp from $FidsTester to $Dns port 53 keep state
pass in on $ext_if inet proto tcp from $FidsTester to $Web port {80,443} keep state
pass in on $ext_if inet proto tcp from $FidsTester to $Mail port 25 keep state
pass in on $ext_if inet proto tcp from $FidsTester to $Ftp port 21 keep state
```

"*Fidstester*" tarafından kullanılacak olan kural tablosu da aşağıda gösterildiği gibi olmaktadır.

```
172.16.10.1:3737:172.16.20.5:23-25:S:TCP
172.16.10.1:3737:172.16.20.6:1-1024:S:TCP
172.16.10.1:3737:172.16.20.7:80:S:TCP
172.16.10.1:3737:172.16.20.7:443:S:TCP
```

```
172.16.10.1:3737:172.16.20.8:1-1024:S:TCP
172.16.10.1:3737:172.16.20.9:21:S:TCP
172.16.10.1:3737:172.16.20.5:53::UDP
172.16.10.1:3737:172.16.20.6:53::UDP
172.16.10.1:3737:172.16.20.7:53::UDP
172.16.10.1:3737:172.16.20.8:53::UDP
172.16.10.1:3737:172.16.20.9:53::UDP
stop_signal=172.16.10.1::172.16.20.3:::ICMP:8:0
```

Belirtilen "*pf*" güvenlik duvarı kural tablosuna ve "*fidstester*" yapılandırma dosyasına göre "*fidstester*" ve "*fidsd*" yazılımları çalıştırıldığında oluşan işlem kayıt dosyaları "*fidsreport*" uygulaması tarafından işlenerek kural tablosu listelenmiştir.

```
Authorized Packets
------------------
1 - 172.16.10.1:3737 > 172.16.20.5:23 S TCP
1028 - 172.16.10.1:3737 > 172.16.20.7:80 S TCP
1029 - 172.16.10.1:3737 > 172.16.20.7:443 S TCP
1054 - 172.16.10.1:3737 > 172.16.20.8:25 S TCP
2054 - 172.16.10.1:3737 > 172.16.20.9:21 S TCP
2055 - 172.16.10.1:3737 > 172.16.20.5:53 UDP
2056 - 172.16.10.1:3737 > 172.16.20.6:53 UDP
2057 - 172.16.10.1:3737 > 172.16.20.7:53 UDP
2058 - 172.16.10.1:3737 > 172.16.20.8:53 UDP
2059 - 172.16.10.1:3737 > 172.16.20.9:53 UDP
Modified Packets
------------------
Dropped Packets
------------------
2 - 172.16.10.1:3737 > 172.16.20.5:24 S TCP
3 - 172.16.10.1:3737 > 172.16.20.5:25 S TCP
…
…
```

“Fidsreport” uygulaması tarafından güvenlik duvarı kural tablosu listelenmiştir. “Dropped Packets” olarak listelenen uzun bir liste aldığı için burada tamamı listelenmemiştir.

Senaryo - 2:

Aşağıda gösterilen "*pf*" kural tablosuna sahip güvenlik duvarı için Şekil 4.18'de gösterildiği gibi içeriden dışarı doğru "*fidstester*" çalıştırılacak ve güvenlik duvarı kural tablosu listelenecektir.

```
ext_if="vic0"
int_if="vic1"
FidsTester="172.16.10.1"
set skip on lo0

nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"
nat on $ext_if from 172.16.20.0/24 -> 172.16.10.2
rdr on $int_if inet proto tcp from 172.16.20.0/24 to any port 21 -> 127.0.0.1
port 8021
anchor "ftp-proxy/*"

block log all
pass out keep state
## Internal -> External
pass in on $int_if proto icmp from 172.16.20.0/24 to $FidsTester keep state
pass in on $int_if proto tcp from 172.16.20.0/24 to $FidsTester port
{22,80,81,82,83,84,85,443,445} keep state
pass in on $int_if proto udp from 172.16.20.0/24 to $FidsTester port
{53,54,55} keep state
```

"*Fidstester*" tarafından kullanılacak olan kural tablosu da aşağıda gösterildiği gibi olmaktadır.

```
172.16.20.4:3737:172.16.10.1:80-90:S:TCP
172.16.20.4:4141:172.16.10.1:443-445:S:TCP
172.16.20.4:2256:172.16.10.1:53-55::UDP
stop_signal=172.16.20.4::172.16.10.1:::ICMP:8:0
```

Belirtilen "*pf*" güvenlik duvarı kural tablosuna ve "*fidstester*" yapılandırma dosyasına göre "*fidstester*" ve "*fidsd*" yazılımları çalıştırıldığında oluşan işlem kayıt dosyaları "*fidsreport*" uygulaması tarafından işlenerek kural tablosu listelenmiştir.

```
Authorized Packets
------------------
Modified Packets
------------------
1 - 172.16.20.4:3737 > 172.16.10.1:80 S TCP
2 - 172.16.20.4:3737 > 172.16.10.1:81 S TCP
3 - 172.16.20.4:3737 > 172.16.10.1:82 S TCP
4 - 172.16.20.4:3737 > 172.16.10.1:83 S TCP
5 - 172.16.20.4:3737 > 172.16.10.1:84 S TCP
6 - 172.16.20.4:3737 > 172.16.10.1:85 S TCP
12 - 172.16.20.4:4141 > 172.16.10.1:443 S TCP
14 - 172.16.20.4:4141 > 172.16.10.1:445 S TCP
15 - 172.16.20.4:2256 > 172.16.10.1:53 UDP
16 - 172.16.20.4:2256 > 172.16.10.1:54 UDP
17 - 172.16.20.4:2256 > 172.16.10.1:55 UDP
Dropped Packets
------------------
7 - 172.16.20.4:3737 > 172.16.10.1:86 S TCP
8 - 172.16.20.4:3737 > 172.16.10.1:87 S TCP
9 - 172.16.20.4:3737 > 172.16.10.1:88 S TCP
10 - 172.16.20.4:3737 > 172.16.10.1:89 S TCP
11 - 172.16.20.4:3737 > 172.16.10.1:90 S TCP
13 - 172.16.20.4:4141 > 172.16.10.1:444 S TCP
```

"*Fidsreport*" uygulaması tarafından güvenlik duvarı kural tablosu listelenmiştir.

BÖLÜM 5. SONUÇ VE ÖNERİLER

Ağ güvenliği konusunun vazgeçilmezleri arasında yer alan güvenlik duvarı ve tehdit gözetleme sistemleri ağ topolojisi içerisinde farklı konumlarda yer alabilirler. Böyle bir topolojinin gerçek olarak modellenmesi için birden fazla sunucu sisteme sahip olunması gerekmektedir. Bu hem maliyeti hem de taşınabilirlik gibi bir takım problemleri beraberinde getirmektedir. Bu çalışmada güvenlik duvarı ve tehdit gözetleme sistemlerinin içerisinde yer aldığı genel bir ağ topolojisi sanallaştırma teknolojisi yazılımı kullanılarak tek bir bilgisayar üzerinde modellenmiştir.

Gerçeklenen bu örnek ağ topolojisi üzerinde güvenlik duvarı ve tehdit gözetleme sistemlerinin işlevselliği ve güvenliğinin test edilmesi işlemi ele alınmıştır. Bu işlemin gerçekleştirilebilmesi için aynı anda birden fazla, farklı farklı yazılımların kullanılması gerekmektedir. Bu çalışma da, bu adımların tek bir uygulama aracılığı ile gerçeklenmesi ilgili yazılım geliştirilmiştir.

Sonuç olarak, tek bir bilgisayar üzerinde gerçeklenen genel bir ağ topolojisi içerisinde, güvenlik duvarı ve tehdit gözetleme sistemlerinin işlevselliği, otomatize olarak tek bir uygulama tarafından başarılı bir şekilde gerçekleştirilmiştir. Aynı zamanda gerçeklenen uygulama gerçek bir ağ topolojisi üzerinde, uygulamanın gerçeklenmesi esnasında sanallaştırma teknolojileri ile oluşturulan topoloji de olduğu gibi, başarılı bir şekilde çalıştırılmış ve uygulanmıştır. Gerçek zamanlı ağ topolojileri üzerinde de uygulanabilir olup çalıştırılabilmektedir.

KAYNAKLAR

[1] DİRİCAN, C.O., Teori ve Uygulamalar ile TCP/IP ve Ağ Güvenliği, ISBN 975-98099-1-5, Açık Akademi, 2005

[2] STEVENS, W. R., TCP/IP Illustrated, Volume 1: The Protocols, ISBN 0-201-63346-9, Addision – Wesley, 1994

[3] GOERZEN, J., Foundations of Python Network Programming, ISBN 1-59059-371-5, Apress, 2004

[4] LUTZ, M., Programming Python, Third Edition, ISBN 10: 0-596-00925-9 ISBN 13: 9780596009250, O'REILLY, 2006

[5] PILGRIM, M., Dive In Python, ISBN-10: 1590593561, ISBN-13: 978-1590593561, APRESS, 2004

[6] The OpenBSD PF Packet Filter Book – Jeremy C. Reed – ISBN-13: 978-0-9790342-0-6 | ISBN-10: 0-97900342-0-5 – Reed Media Services, 2006

[7] BURNS, B., GRANICK, J., MANZUIK, S., GUERSCH, P., KILLION, D., BEAUCHESNE, N., MORET, E., SOBRIER, J., LYNN, M., MARKHAM, E., IEZZONI, C., BIONDI., P., Security Power Tools, ISBN 10: 0-596-00963-1, ISBN 13:9780596009632 - O'REILLY, 2007

[8] BRIAN Caswell, Jay Beale, ANDREW R Baker, Snort IDS and IPS Toolkit, ISBN-10: 1597490997, ISBN1-13: 978-1597490993 – Syngress, 2007

ÖZGEÇMİŞ

Gökhan ALKAN, 01.02.1980 ‘de Kastamonu’da doğdu. İlk, orta ve lise eğitimini Kastamonu’da tamamladı. 2004 yılında Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümünden mezun oldu. 2005 - 2007 yılları arasında Kocaeli Üniversitesi Bilgi İşlem Daire Başkanlığı’nda Linux/Unix sistem yöneticisi olarak çalıştı. 2007 - 2009 yılları arasında TÜBİTAK/UEKAE’ de araştırmacı olarak görev yaptı. Bu süre zarfı içerisinde başta Linux/Unix sistem yöneticiliği ve ağ güvenliği konuları üzerine çalışmalarda bulundu. Gökhan ALKAN, halen aynı görevine devam etmektedir.