T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM TANIMLI AĞLARIN VE GELENEKSEL BİLGİSAYAR AĞLARININ GÜVENLİK KARŞILAŞTIRMASI

ABDULLAH YAVUZ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Prof. Dr. Gürkan TUNA

EDİRNE-2020

Abdullah YAVUZ'un hazırladığı "Yazılım Tanımlı Ağların ve Geleneksel Bilgisayar Ağlarının Güvenlik Karşılaştırması" başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Bilgisayar Mühendisliği Anabilim Dalında bir **Yüksek Lisans tezi** olarak kabul edilmiştir.

Jüri Üyeleri (Ünvan, Ad, Soyad): İmza

Prof. Dr. Gürkan TUNA

Dr. Öğr. Üyesi Tarık YERLİKAYA

Dr. Öğr. Üyesi Bora ASLAN

Tez Savunma Tarihi: 04/09/2020

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

İmza

Prof. Dr. Gürkan TUNA
Tez Danışmanı

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

Prof. Dr. Murat YURTCAN
Fen Bilimleri Enstitüsü Müdürü

T.Ü. FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI YÜKSEK LİSANS PROGRAMI

DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

04/09/2020

Abdullah YAVUZ

Yüksek Lisans Tezi

Yazılım Tanımlı Ağların ve Geleneksel Bilgisayar Ağlarının Güvenlik Karşılaştırması

T.Ü. Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

Bu tez çalışmasında yazılım tanımlı ağlar ve geleneksel bilgisayar ağlarının güvenlik karşılaştırılması gerçekleştirilmiştir. Bu karşılaştırmada Hping3 ve Dsniff uygulamaları kullanılmış ve gerçekleştirilen benzetim çalışmalarında IP Spoofing, SYN Flood, RST/FIN Flood, SYN-ACK Flood, UDP Flood, ARP Poisoning ve Distributed Denial of Service saldırıları gerçekleştirilmiştir. Yazılım tanımlı ağ oluşturmak için Ubuntu 14.04 LTS sanal makinesi üzerinde Floodlight yazılım tanımlı ağ kontrolcüsü, ağ topolojisini oluşturmak için ise Mininet kullanılmıştır. Geleneksel bilgisayar ağlarındaki benzetim çalışması yapmak için ise Kali Linux sanal makinesi kullanılmıştır. Geleneksel bilgisayar ağlarında siber saldırılar ana bilgisayara yapılmıştır.

Gerçekleştirilen benzetim çalışmalarının sonuçları, geleneksel bilgisayar ağlarına kıyasla yazılım tanımlı ağların daha fazla güvenlik zafiyeti taşıdığı göstermektedir. Öte yandan, alınacak bazı tedbirler yazılım tanımlı ağlardaki bu güvenlik zafiyetlerini ortadan kaldırılabilir.

Yıl : 2020

Sayfa Sayısı : 68

Anahtar Kelimeler : yazılım tanımlı ağ, geleneksel bilgisayar ağları, güvenlik

Master Thesis
Security Comparison of Software Defined Networks and Traditional Computer Networks
Trakya University Institute of Natural Sciences
Computer Engineering Department

ABSTRACT

In this thesis, the security comparison of software defined networks and traditional computer networks was carried out. Hping3 and Dsniff applications were used in this comparison and IP Spoofing, SYN Flood, RST / FIN Flood, SYN-ACK Flood, UDP Flood, ARP Poisoning and Distributed Denial of Service attacks were performed in the simulation studies. Floodlight software defined network controller was used on Ubuntu 14.04 LTS virtual machine to create a software defined network, and Mininet was used to create the network topology. The Kali Linux virtual machine was used to simulate a traditional computer network. In the traditional computer network, cyber attacks were made to the host computer.

The results of the simulation studies show that software-defined networks have higher security vulnerabilities compared to traditional computer networks. On the other hand, some measures to be taken can eliminate these vulnerabilities in software defined networks.

Year : 2020
Number of Pages : 68
Keywords : software defined networks, traditional computer networks, security

TEŞEKKÜR

Yüksek lisans eğitimim boyunca bana destek olan, bilgi ve tecrübesiyle çalışmalarımın her aşamasında bana destek olan saygıdeğer danışmanım Prof. Dr. Gürkan TUNA'ya teşekkürlerimi sunarım.

Yüksek Lisans öğrenimim boyunca her zaman destek olan ve anlayışlı davranan çalışma arkadaşlarıma ve tüm öğrenim hayatım boyunca ve her zaman bana destek olan aileme teşekkür ederim.

İÇİNDEKİLER

ÖZET iv
ABSTRACT v
TEŞEKKÜR vi
İÇİNDEKİLER vii
ŞEKİLLER LİSTESİ ix
ÇİZELGE LİSTESİ x
KISALTMALAR xi
GİRİŞ 1
GELENEKSEL BİLGİSAYAR AĞLAR 3
2.1. OSI Referans Modeli 4
2.1.1. Fiziksel Katman 4
2.1.2. Veri Bağlantı Katmanı 5
2.1.3. Ağ Katmanı 5
2.1.4. Taşıma Katmanı 5
2.1.5. Oturum Katmanı 5
2.1.6. Sunum Katmanı 6
2.1.7. Uygulama Katmanı 6
2.2. TCP/IP Modeli 6
2.3. Ağ Protokolleri 8
2.3.1. İnternet Protokolü 8
2.3.2. Ethernet 8
2.3.3. Fiziksel Adres 8
2.3.4. Adres Çözümleme Protokolü 8
2.3.5. İletim Kontrol Protokolü 9
2.3.6. Kullanıcı Veri Bloğu Protokolü 10
2.4. Alan Ağları 11
2.4.1. Kişisel Alan Ağı 11
2.4.2. Yerel Alan Ağı 11
2.4.3. Kampus Alan Ağı 12
2.4.4. Büyükşehir Alan Ağı 12
2.4.5. Geniş Alan Ağı 12
2.5. Ağ Cihazları 12
2.6. Geleneksel Bilgisayar Ağlarında Güvenlik 13

YAZILIM TANIMLI AĞLAR ve MİNİNET 15
3.1 Yazılım Tanımlı Ağlar 15
3.1.1. OpenFlow Protokolü 17
3.1.2. Yazılım Tanımlı Ağ Mimarisi 17
3.1.3. Yazılım Tanımlı Ağ Kontrolcüleri 20
3.1.4. Yazılım Tanımlı Ağlarda Güvenlik 23
3.2. Mininet 25
3.2.1 Mininet Topolojileri 27
3.2.2. Mininet Ağı Oluşturmak İçin Kullanılan Komutlar 29
3.2.3. Mininet CLI'da Kullanılan Basit Komutlar 29
MATERYAL METOD 31
4.1. Test Ortamlarının Hazırlanması 31
4.2. Benzetim Çalışmalarında Kullanılan Saldırılar 34
4.2.1. IP Spoofing 35
4.2.2.SYN Flooding 35
4.2.3. RST/FIN Flood 36
4.2.4. SYN-ACK Flood 37
4.2.5. UDP Flood 38
4.2.6. DDoS Saldırısı 38
4.2.7. ARP Poisoning Saldırısı 39
4.3. Saldırıların Yapılışı 40
SONUÇLAR VE TARTIŞMA 41
5.1. IP Spoofing Saldırısı Sonuçları 41
5.2. SYN Flooding Saldırısı Sonuçları 42
5.3. RST/FIN Flood Saldırısı Sonuçları 44
5.4. SYN-ACK Flood Saldırısı Sonuçları 45
5.5. UDP Flood Saldırısı Sonuçları 47
5.6. DDOS Saldırısının Sonuçları 48
5.7. ARP Poisoning Saldırısı Sonuçları 50
5.8. Değerlendirme 51
KAYNAKLAR 52
ÖZGEÇMİŞ 56
TEZ ÖĞRENCİSİNE AİT TEZ İLE İLGİLİ BİLİMSEL FAALİYETLER 57

ŞEKİLLER LİSTESİ

Şekil 2.1. OSI Referans Modeli Katmanları 4
Şekil 2.2. TCP/IP Modeli Katmanları 7
Şekil 2.3. TCP Üçlü El Sıkışma Yöntemi 9
Şekil 2.4. Alan Ağlarının Karşılaştırmalı Gösterimi 12
Şekil 3.1. ONF'nin Önerdiği YTA Mimarisi 18
Şekil 3.2. Python Kullanarak Oluşturulan Özel Topoloji 28
Şekil 3.3. Mininet CLI'da Kullanılan Basit Komutlar 29
Şekil 4.1. Mininet Yardımıyla Oluşturulan Ağın Topolojisi 34
Şekil 5.1. IP Spoofing Saldırı Sonuçları 42
Şekil 5.2. SYN Flooding Saldırısı Sonuçları 43
Şekil 5.3. RST/FIN Flood Saldırısı Sonuçları 45
Şekil 5.4. SYN-ACK Flood Saldırısı Sonuçları 46
Şekil 5.5. UDP Flood Saldırısının Sonuçları 48
Şekil 5.6. DDoS Saldırısı Sonuçları 49
Şekil 5.7. ARP Poisoning Saldırısının Sonuçları 51

ÇİZELGE LİSTESİ

Çizelge 2.1. OSI ve TCP/IP Modelinin Karşılaştırmalı Gösterimi 7
Çizelge 2.2. TCP ve UDP'nin Özelliklerinin Karşılaştırması 11
Çizelge 3.1. ONF ve Diego Kreutz Önerdiği Mimarilerin Karşılaştırmalı Gösterimi 19
Çizelge 3.2. YTA Kontrolcü Özellikleri (Salman, Elhajj, Kayssi ve Chehab, 2016) 22
Çizelge 3.3. Tehdit Vektörlerinin Açıklaması 24

KISALTMALAR

YTA	Yazılım Tanımlı Ağ
ONF	Open Networking Foundation
IEEE	Institute of Electrical and Electronics Engineers
DoS	Denial of Service
DDoS	Distributed Denial of Service
OSI	Open System Interconnection
ISO	International Organization for Standardization
IP	Internet Protocol
ARP	Address Resolution Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
PAN	Personal Area Network
LAN	Local Area Network
CAN	Campus Areas Network
MAN	Metropolitan Area Network
WAN	Wide Area Network
SNMP	Simple Network Managment Protocol
FTP	File Transfer Protocol
CLI	Command Line Interface

BÖLÜM 1

GİRİŞ

Bilgisayarlar arasındaki haberleşmenin kolaylaşması için bilim insanları tarafından farklı protokoller geliştirilmiştir. Fakat geliştirilen bu protokoller, bilgisayar ağlarında yeni sorunlar yaratmaya başlamıştır. Her yeni geliştirilen protokol bir sorunu çözerken başka sorunlar oluşmasına neden olmuştur. Geleneksel bilgisayar ağları diye adlandırılan bu ağlar, önceden tanımlı ağ politikalarına göre yapılandırılır. Bu durum geleneksel bilgisayar ağlarının kontrol edilmesini zorlaştırmaktadır (Kreutz, Ramos, Verissimo, Rothenberg, Azodolmolky & Uhlig, 2015). Geleneksel bilgisayar ağları kontrol edilebilirlik, esneklik, performans, tasarım ve daha birçok konuda yeniliğe ihtiyaç duymaktadır. Bu durum bizi, yazılım tanımlı ağ (YTA) olarak bilinen yeni nesil ağlara yönlendirmektedir.

Geleneksel bilgisayar ağları karmaşık ve yönetilmesi zordur. Ağ politikalarını oluşturmak için, ağ operatörlerinin her bir ağ cihazını satıcıya özgü komutlar kullanarak ayrı ayrı yapılandırması gerekmektedir. Otomatik olarak yeniden yapılandırma geleneksel bilgisayar ağlarında bulunmamaktadır. Bu nedenle geleneksel bilgisayar ağlarında dinamik ağ politikalarını uygulamak zordur. Yazılım tanımlı ağlar, geleneksel bilgisayar ağlarının aksine oluşturulan ağ politikalarını dinamik olarak uygulanmasını sağlar. (Kreutz vd., 2015)

YTA'ları geleneksel bilgisayar ağlarından ayıran 2 tane karakteristik özellik vardır. Bunlardan birincisi veri düzlemi ve kontrol düzleminin birbirinden ayrılmasıdır. Diğeri ise, ağ uygulamaları geliştirmek için programlanabilirlik sağlanmasıdır. Programlanabilirlik sayesinde YTA, daha yenilikçi ağ tasarımları, ağı daha etkili

yapılandırma olanağı, daha iyi performans ve ağda esneklik olanağı sağlamaktadır (Sezer, Scott-Hayward, Chouhan, Fraser, Lake, Finnegan & Rao, 2013).

Geleneksel bilgisayar ağlarının karmaşıklığının ve kontrol edilebilirliğinin yanı sıra güvenlik tarafında da büyük zafiyetleri vardır. YTA aynı zamanda geleneksel bilgisayar ağlarının güvenlik sorunları ile ilgili çözümler sunmaktadır. Fakat bu güvenlik sorunları şu anda yüzde yüz çözülememiştir. Ayrıca veri düzlemi ve kontrol düzleminin ayrılmasının sonucunda YTA'da yeni güvenlik sorunları baş göstermiştir. ARP Spoofing, servis reddi (Denial of Service (DOS)) saldırısı ve veri ihlali gibi saldırılar şu anda YTA'nın maruz kaldığı saldırı türlerinden bazılarıdır.

YTA'ların maliyeti, ağ tasarımı, yönetilebilir olması gibi özellikler önemlidir. Fakat bu özelliklerin öneminin daha da artması için YTA'nın güvenlik sorunları iyi bir şekilde çözülmüş olmalıdır. Geleneksel bilgisayar ağlarda bulunan güvenlik açıklarının yanı sıra YTA'nın veri katmanı ve düzlem katmanını ayrılması ile yeni birçok güvenlik problemi ortaya çıkmıştır.

Bu çalışmada YTA oluşturmak için, Ubuntu 14.04 sanal makinesi içerisinde Floodlight YTA kontrolcüsünü ve topolojileri oluşturmak için Mininet emülatörü kullanılmıştır. Çalışmanın amacı, günümüzde kullanılan geleneksel bilgisayar ağları ile YTA'nın güvenlik karşılaştırmalarını analiz etmektir.

BÖLÜM 2

GELENEKSEL BİLGİSAYAR AĞLAR

Bir geleneksel bilgisayar ağı, düğümlerin kaynakları paylaşmasını sağlayan dijital bir telekomünikasyon ağıdır. Geleneksel bilgisayar ağlarında, bilgisayar cihazları, düğümler arasındaki bağlantıları kullanarak birbirleriyle veri alışverişinde bulunur. Bu veri bağlantıları, teller veya optik kablolar gibi kablolu ortamlar veya kablosuz ortamlar üzerinden kurulur.

Verileri alan, yönlendiren ve sonlandıran bilgisayar cihazlarına ağ düğümleri denir. Düğümler genellikle ağ adresleriyle tanımlanır ve kişisel bilgisayarlar, telefonlar ve sunucular gibi ana bilgisayarları ve ayrıca yönlendirici ve anahtarlayıcı gibi ağ donanımlarını içerebilir. Bu tür iki cihazın, bir cihaz diğer cihazla bilgi alışverişinde bulunabildiği zaman, birbirlerine doğrudan bağlantısı olup olmadıklarına bağlı olarak söylenebileceği söylenebilir. Çoğu durumda, uygulamaya özel iletişim protokolleri, diğer daha genel iletişim protokolleri üzerinden katmanlanır. Bu müthiş bilgi teknolojisi koleksiyonu, güvenilir bir şekilde çalışmasını sağlamak için yetenekli ağ yönetimi gerektirir.

Geleneksel bilgisayar ağları, genellikle çok sayıda göbek, yönlendirici ve anahtarlayıcı gibi ağ cihazlarından ve bunlar üzerinde uygulanan çok karmaşık protokollerden oluşmaktadır. Ağ operatörleri, çok çeşitli ağ olaylarına ve uygulamalarına yanıt verecek politikaları yapılandırmaktan sorumludur. Değişen ağ koşullarına adapte olurken bu yüksek seviye politikalarını manuel olarak düşük seviye yapılandırma komutlarına dönüştürmeleri gerekir. Sık sık bu çok karmaşık görevleri çok sınırlı araçlara

erişerek gerçekleştirmeleri gerekir. Sonuç olarak, ağ yönetimi ve performans ayarlama oldukça zordur ve bu nedenle hataya açıktır.

2.1. OSI Referans Modeli

Open System Interconnection (OSI) referans modeli International Organization for Standardization (ISO) tarafından geliştirilmiştir. Buradaki amaç iki bilgisayar arasındaki iletişimin nasıl olacağını tanımlamaktır. OSI öncesindeki dönemde, bilgisayarın donanım parçalarını üreten şirketlere özgü ağlar bulunmaktaydı. Bu ağlar, kendi üreticisinin donanımının bağlanmasına imkân sağlayacak biçimde tanımlanmıştı. OSI, çeşitli bilgisayar donanımı üreten şirketlerin bağlanabileceği bir ağ modeli olarak ortaya çıktı. OSI referans modeli herhangi bir donanıma göre farklılık göstermeden ortak bir mimarinin ve protokollerin bir ağ bileşeni gibi kullanılmasını sağlamaktadır. OSI referans modeli birbirinden farklı görevlere sahip 7 katmana ayrılmıştır. Bu katmanlar Şekil 2.1'de gösterilmektedir.



Şekil 2.1. OSI Referans Modeli Katmanları (ITU-T Rec. X.200, 1994)

2.1.1. Fiziksel Katman

Fiziksel katman bilgisayarlar arasındaki iletişimin nasıl yapılacağına karar vermektedir. Veriler bit olarak iletilir. Bitler bu katmanda elektrik, ışık veya radyo

sinyallerine çevrilerek gönderilir. Alıcı taraf, kendisine gönderilen iletim türünü tekrardan bir ve sıfırlara dönüştürür. Verilerin iletiminin mümkün olması için hem gönderici hem de alıcı tarafta aynı kurallar tanımlanmış olmalıdır.

2.1.2. Veri Bağlantı Katmanı

Veri bağlantı katmanı fiziksel katmana erişmek için kullanılan kuralların belirlendiği katmandır. Bu erişim yöntemleri verileri kendi protokollerine göre işleyip fiziksel katmana iletirler. Burada bulunan veriler çerçeve denen parçalara bölünmektedir. Çerçeveler, paketlerin belli bir kontrol içinde gönderilmesini sağlayan paketlerdir. Veri bağlantı katmanı ağdaki diğer cihazları tanıma ve fiziksel katmanda gelen verilerin hata kontrolünün yapıldığı katmandır.

2.1.3. Ağ Katmanı

Ağ katmanı, gönderilecek olan verilerin gideceği ağın bilgisini ve gerektiğinde yönlendiricilerin kullanacağı bilginin eklendiği katmandır. Veriler paketlere bölünür. Ağ katmanı verilerin iletilebilecek en ekonomik yoldan gönderilmesine karar verir. Bu katman sayesinde veriler yönlendiriciler aracılığıyla gönderilir. Ağ katmanında veriler mantıksal adresler olan MAC adreslerine dönüştürülür. Yönlendirme işlemi bu aşamada yapılmaktadır. IP protokolünün çalıştığı katman burasıdır.

2.1.4. Taşıma Katmanı

Taşıma katmanı üst katmanlardan gelen veriyi parçalara bölerek ağ paketi boyutuna dönüştürür. İletim Kontrol Protokolü (Transmission Control Protocol (TCP)) ve Kullanıcı Veri Bloğu Protokolü (User Datagram Protocol (UDP)) protokolleri burada çalışmaktadır. Veriler kesim halinde taşınır. Hata kontrolünün yapıldığı katman burasıdır. Alt katmanlardan gelen veriyi üst katmanlara taşır ve servis kalitesini arttırır. Verilerin zamanında iletilip iletilmediğini kontrol eder ve uçtan uca iletilmesini sağlar.

2.1.5. Oturum Katmanı

Oturum katmanında iki bilgisayar arasında bağlantının yapılması kullanımı ve bitirilmesi işlemleri yapılmaktadır. Bir bilgisayarın birden fazla bilgisayarla iletişim halinde olmasından dolayı gerektiğinde bilgisayarlarla konuşmasını sağlar.

2.1.6. Sunum Katmanı

Sunum katmanı gönderilecek verilerin hedef bilgisayar tarafından anlaşılacak şekilde dönüştürüldüğü katmandır. Bu özelliği sayesinde farklı programların birbirlerinin verisini kullanılmasını mümkün kılar. Uygulama katmanına gönderilecek verilerin yapısı ve formatının belirlendiği yerdir. Verilerin sıkıştırılması, şifrelenmesi ve açılması bu katmanda yapılır.

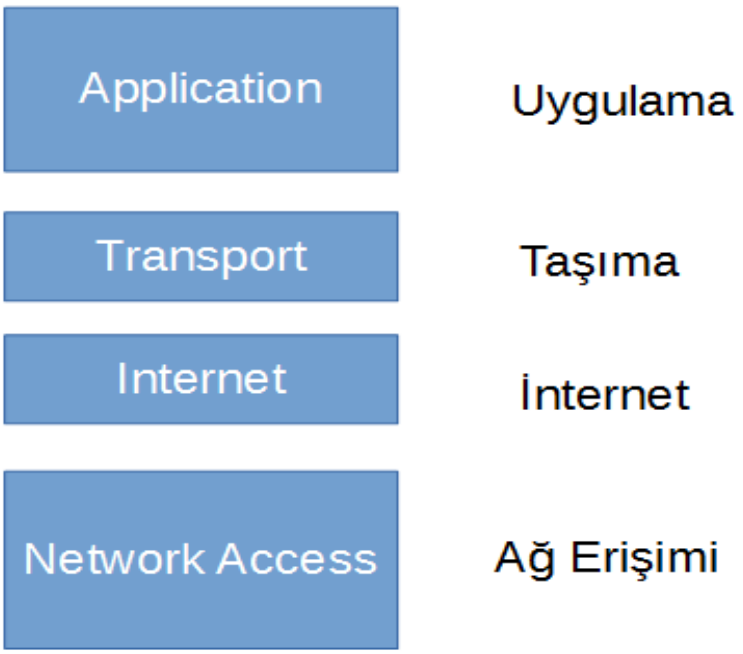
2.1.7. Uygulama Katmanı

Bilgisayar uygulaması ile ağ birimi arasında iletimin yapıldığı katmandır. OSI katmanları arasında sadece bu katman diğer katmanlara servis sağlamaz. Uygulamaların ağın üzerinde çalışmasını sağlar.

2.2. TCP/IP Modeli

Amerikan Savunma Bakanlığı'nın farklı yerlerdeki bilgisayarların haberleşmesi projesi olarak ortaya atıldı. İlk olarak 1969 yılında 4 farklı yerdeki bilgisayarların birbirlerine bağlanmasıyla ARPANET adı verilen bir ağ oluşturulmuştur. Üniversiteler ve araştırma merkezleri daha sonrasında bu projeye dahil olmuştur. En sonunda tüm kullanıcıların dahil olduğu ağ olan İnternet ortaya çıkmıştır (Taner, 2018).

OSI referans modeli gibi TCP/IP modeli açık standarttır. Birçok alt katman protokolleri bulunmaktadır. Bunlar Institute of Electrical and Electronics Engineers (IEEE) tarafından geliştirilmektedir. Şekil 2.2'de görüldüğü gibi 4 katmandan oluşmaktadır. OSI ve TCP/IP modelinin katman karşılaştırması Çizelge 2.1'de gösterilmiştir.



Şekil 2.2. TCP/IP Modeli Katmanları (RFC 1180, 1991)

Çizelge 2.1. OSI ve TCP/IP Modelinin Karşılaştırmalı Gösterimi

OSI Referans Modeli	**TCP/IP Modeli**
Uygulama Katmanı	Uygulama Katmanı
Sunum Katmanı	
Oturum Katmanı	
Taşıma Katmanı	Taşıma Katmanı
Ağ Katmanı	İnternet Katmanı
Veri Bağlantı Katmanı	Ağ Erişimi Katmanı
Fiziksel Katmanı	

2.3. Ağ Protokolleri

2.3.1. İnternet Protokolü

İnternet Protokolü (IP), verilerin internet üzerinden bir cihazdan diğerine gönderilmesini sağlayan protokoldür. İnternette bulunan her cihazın benzersiz bir biçimde tanımlanan IP adresi vardır. İnternet üzerinden gönderilecek veriler küçük parçalara bölünmektedir. Bu parçalar hem gönderenin hem de alıcının IP adresine sahiptir.

2.3.2. Ethernet

Ethernet bilgisayarlar ve ağ cihazlarının kablolu veya kablosuz olarak birbirlerine bağlanmasını sağlayan protokoldür. OSI referans modelinde veri bağlantı ve fiziksel katmanda çalışmaktadır. MAC adresleri üzerinden haberleşme standardı ve ortak bir adresleme formatı tanımlar. Günümüzde yerel alan ağlarının altyapısını oluşturan karmaşık ağ teknolojisi yapısına evrilmiştir.

2.3.3. Fiziksel Adres

Fiziksel (Medium Access Control (MAC)) adresler yerel ağlarda kullanılmak için üretilmiş 48 bitten oluşur ve her cihaza benzersiz olarak atanır. Bu adresler 16'lık tabanda 12 haneden oluşmaktadır. IEEE tarafından üretilmiş bu adresler, ilk 6 hanesi üreticinin kimliğini belirtir ve bütün cihazlarda aynıdır. Son 6 hanesi ise MAC adreslerinin benzersiz olmasını sağlar.

2.3.4. Adres Çözümleme Protokolü

Ethernet yerel alan ağlarında en çok kullanılan arayüzdür. Bu arayüzler 48 bitlik MAC adresleri üzerinden birbirleri ile haberleşmektedir. TCP/IP ise haberleşmek için 32 bitlik IP adreslerini kullanmaktadır. Yerel alan ağında cihazlar birbirleri ile haberleşmek için ağda bulunan cihazların MAC adreslerini bilmek zorundadır. Ağda bulunan cihazların MAC adreslerini öğrenme işlemini Adres Çözümleme Protokolü (Address Resolution Protocol (ARP)) sağlamaktadır.

Bir bilgisayar IP'sini bildiği fakat MAC adresini bilmediği bir bilgisayar varsa bütün ağa ARP isteği (ARP request) gönderilir. Bu pakette göndericinin IP adresi MAC adresi ve alıcı bilgisayarın ise IP adresi bulunmaktadır. Bu pakete cevap hedef IP adresinden gelir. Bu cevap pakette hedef bilgisayarın MAC adresi yer alır. İsteği yollayan ve isteği

yanıtlayan bilgisayar birbirlerinin IP ve MAC adreslerini daha sonra kullanmak üzere ARP tablolarına kaydeder. ARP paketi isteğinin başlangıcı ile bilgisayarların bilgilerinin belleklerine kaydetme işlemine ARP MAC adreslerinin çözümlenmesi denir.

ARP işlemini tekrar yapmamak için bilgisayarların öğrenilen MAC adresi ve IP adresi bilgilerini ileride kullanmak için kaydettikleri yerlere ARP tablosu denmektedir. ARP tablosu otomatik olacağı gibi el ile de doldurulabilir. ARP girdilerinin, ARP tablolarında 10 dakikalık ömre sahiptir. Tabloya MAC adresi ve IP adresi bilgileri girildikten sonra 2 dakika içinde kullanılmayan bilgiler tablodan atılır.

2.3.5. İletim Kontrol Protokolü

TCP/IP modelinin taşıma katmanında yer alan protokoldür. Gönderilen paketlerin doğru sıra ve istenilen rota ile gönderilmesini sağlamak için kullanılır. İki bilgisayar arasında bağlantıyı sağlar. Akış kontrolü yaparak gönderilen verilerin doğru bir şekilde ulaşıp ulaşmadığını kontrol eder.

İletim Kontrol Protokolü iki bilgisayar arasında haberleşmeyi başlatmak için üçlü el sıkışma denilen bir yöntem uygular. Kaynak bilgisayar ilk olarak rastgele bir sıra numarası üreterek hedef bilgisayara SYN mesajı yollamaktadır. SYN mesajını alan hedef bilgisayar, kaynak bilgisayara SYN ve ACK (ACKnowledgement) mesajı yollar. Kaynak bilgisayar en sonunda hedef bilgisayara aralarındaki bağlantının kurulduğunu bildiren bir ACK (TCP connection is ESTABLISHED) mesajını alır. Şekil 2.3'te üçlü el sıkışma yöntemi gösterilmektedir.



Şekil 2.3. TCP Üçlü El Sıkışma Yöntemi

TCP kaynak bilgisayara verileri gönderdikten sonra kontrol paketi gönderir. Bu pakette gönderilen paketlerin hangilerinin gönderildiği ve ne kadarının doğru alındığı bilgisini içermektedir. Kaynak bilgisayara kontrol paketi gelmez ise kaynak bilgisayar birkaç daha paketleri gönderir. Böylece paketlerin kesin olarak gönderildiğinden emin olur.

2.3.6. Kullanıcı Veri Bloğu Protokolü

Kullanıcı Veri Bloğu Protokolü, TCP'ye alternatif bir protokoldür. TCP'nin aksine gecikmenin az ve kayıplara dayanıklı iletimlerde kullanılır. UDP datagram adı verilen mesajları iletir ve en çok çaba gerektiren protokol olarak kabul edilmektedir. TCP'nin aksine hata ve akış kontrolü sağlamaz. UDP bağlantı gerektirmeyen bir protokoldür.

UDP farklı kullanıcıların isteklerini ayırt etmek için port numaralarını ve isteğe bağlı olarak verilerin eksiksiz geldiğini kontrol etmek için bir sağlama toplama özelliği bulunmaktadır. UDP elinde bulunan paketleri sadece göndermektedir. Gönderilen paketler, hedef adres farklı yollardan gidebilir ve bazen gönderilen paketlerde kayıplar olmaktadır. Genellikle oyun, sesli veya görüntülü ve akışlı ortam gibi hata düzeltmenin gerekli olmadığı yerlerde kullanılabilir. Çizelge 2.2'de TCP ve UDP arasındaki farklar gösterilmektedir.

Çizelge 2.2. TCP ve UDP'nin Özelliklerinin Karşılaştırması

Servis	TCP	UDP
Bağlantı Kurulumu	Güvenilir bağlantı kurulur.	Güvenilir bağlantı kurulmaz.
Paketlerin Sırası Hakkında Bilgi	Paketler gönderilirken ardışık sıralanır.	Ardışık paket numarası yoktur.
Akış Kontrolü	Alıcı vericiye yavaşlaması için sinyal gönderebilir.	Akış kontrolü için TCP'de kullanılan onay UDP'den geri dönmez.
Tıkanıklık Kontrolü	TCP onay paketleri kullanarak ağ cihazları göndericinin tavrını kontrol edebilir.	Ağ tıkanıklığı sinyali göndermez.
Teslim Garantisi	Paketlerin alıcıya gönderildiğini onaylar.	UDP'de paketlerin teslim garantisi yoktur. Kaybolan paketler bir daha gönderilmez.

2.4. Alan Ağları

2.4.1. Kişisel Alan Ağı

En küçük ve en temel ağ (Personal Area Network (PAN)) türüdür. Bu ağ, kablosuz bir modem bir veya iki bilgisayardan, tabletlerde vb. oluşur. Genellikle bir kişi etrafında döner. Küçük ofis ve konutlarda bulunur.

2.4.2. Yerel Alan Ağı

En basit, en yaygın ve en özgün alan ağlarından (Local Area Network (LAN)) biridir. Bilgi kaynaklarını paylaşmak için bilgisayar gruplarını kısa mesafelerde birbirine bağlamaktadır. İşletmeler yerel alan ağlarını yönetir ve bakımlarını yapar.

2.4.3. Kampus Alan Ağı

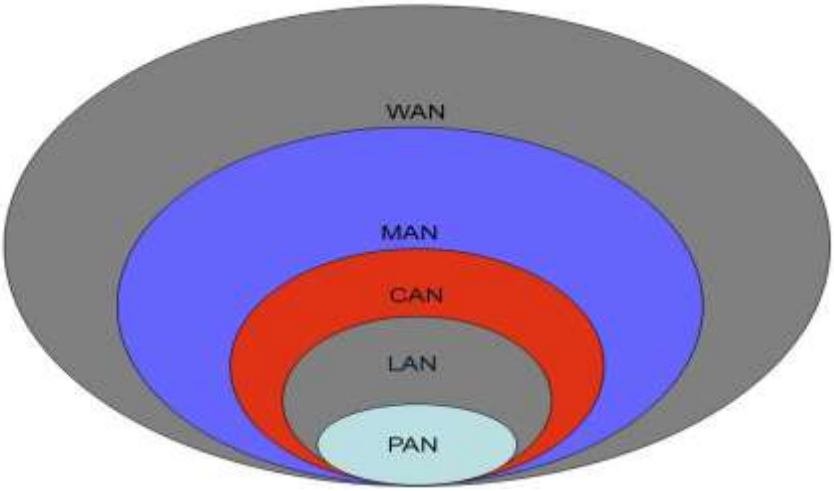
Yerel alan ağlarından büyüktür (Campus Area Network (CAN)). Genellikle üniversite ve okul bölgelerinde bulunmaktadır.

2.4.4. Büyükşehir Alan Ağı

Bütün bir coğrafi bölgeyi kapsar (Metropolitan Area Network (MAN)). Yerel alan ağlarından büyük, geniş alan ağlarından küçüktür. Her iki ağ türünden öğeler içermektedir.

2.4.5. Geniş Alan Ağı

En karmaşık yapıya sahip ağ türüdür (Wide Area Network (WAN)). Bilgisayarlar yerel alan ağına göre daha uzun mesafeden birbirlerine fiziksel olarak bağlıdır. Şekil 3.4'te alan ağlarının boyutları karşılaştırmalı olarak gösterilmiştir.



Şekil 2.4. Alan Ağlarının Karşılaştırmalı Gösterimi

2.5. Ağ Cihazları

İki veya daha fazla bilgisayarı birbirine bağlamak için ilave donanım cihazlarına ihtiyaç duyulmaktadır. Ayrıca bilgisayarlar arasındaki mesafe arttıkça gönderilen sinyal gücünün zayıflaması ile karşılaşılır. Bu durumlarda ağı destekleyecek donanımlara ihtiyaç duyulur. Bu donanımlara ağ cihazları denir. Bu cihazlar OSI referans modeline göre farklı katmanlarda çalışmaktadır.

- Göbek (Hub)

En basit ağ cihazıdır. Kendine ait bir güç kaynağı vardır. Çok portlu olması sayesinde ağ elemanlarının birbirine bağlanmasını sağlar. Kendisine gelen veriyi bütün portlara gönderir. Ağda sinyallerin yeniden oluşturulmasını ve yeniden zamanlanmasını sağlar. OSI referans modelinde 1. katmanda çalışmaktadır. Göbekler birbirine bağlanarak ağın genişlemesini sağlar.

- Anahtarlayıcı (Switch)

Kendine bağlı bilgisayara anahtarlamalı olarak yol sunar. Anahtarlama özelliğinden dolayı göbek cihazlarına göre daha performansı daha yüksektir. 8 ile 48 arasında port sayısına sahip modelleri vardır. OSI referans modelinde 2. katmanda çalışmaktadır. Paketleri MAC adresine göre yönlendirir.

- Köprü (Bridge)

Aynı protokolü kullanan birbirinden bağımsız iki ağın arasına konur. Veri adresi ağdaki adreslerden biri ile eşleşiyorsa geçmesine izin verir aksi takdirde verinin geçişine izin vermez.

- Yönlendirici (Router)

Uzak ağa erişmek için kullanılır. Verileri gönderirken en iyi yol seçimini yapıp öyle gönderir. OSI referans modelinde 3. seviyede çalışmaktadır. Sadece ağ adresi bilinen verilerin aktarılmasına izin vermesinden dolayı ağ trafiği azdır.

2.6. Geleneksel Bilgisayar Ağlarında Güvenlik

Siber güvenlik, verileri, programları, aygıtları ve ağları, saldırı veya yetkisiz erişime karşı korumak için tasarlanmış uygulamalar, teknolojiler ve süreçler olarak ifade edilebilir. Başka bir tanımda bilgisayar bilgi sistemlerine, alt yapılarına, ağlarına ya da kişisel bilgisayar cihazlarına veri ya da bilgi sistemlerini çalmak veya yok etmek amacıyla yapılan herhangi saldırgan aksiyona siber saldırı denmektedir Bilgi güvenliği alanında siber güvenlik önemli faktör haline gelmiştir. Siber güvenlikte bilgi güvenliği, siber saldırılardan doğacak zararları en aza indirmeyi amaçlar.

Siber güvenliği nesneleri yetkisiz erişimlerden korumak için sürekli devam eden süreç olarak tanımlar. Bu nesne, bir kişi, iş gibi bir kuruluş ya da bilgisayar sistemi veya dosya olabileceği belirtilmiştir (Kizza, 2009). Bir bilgisayar sistemine, altyapıya, ağa veya akıllı cihazlara devre dışı bırakma, yok etme veya yetkisiz erişim elde etme girişimine siber saldırı denir. Kullandığı saldırı tekniğinden ve amaçlarından dolayı farklı türlerde siber saldırılar bulunmaktadır. Servis reddi, ortadaki adam (man-in-the-middle) ve zararlı yazılım saldırısı başlıca kullanılan siber saldırılardır.

- Servis Reddi (Denial of Service (DoS))

Servis reddi saldırısı, bir sistemin kaynaklarını hizmet isteklerine yanıt vermesini engellemek için yapılan saldırıdır. Saldırganın sisteme doğrudan erişim hakkı kazandırmaz. Sadece sistemin kaynaklarını tüketmesini sağlayarak sistemin kullanılamaz olmasına neden olur. Kullandığı zaafiyete göre farklı türlerde servis reddi saldırı mevcuttur.

- Ortadaki Adam (Man-in-the-middle)

Ortadaki adam saldırısında saldırgan kendini aralarında iletişim olan iki tarafın arasına kendini gizlice sokup aralarındaki haberleşmeye dahil olmaktadır. Saldırgan iletişim trafiğini kaynaktan keser ve hedefe iletir, böylece iletileri değiştirme ve taraflardan herhangi birinin farkına varmadan yenilerini ekleme yeteneği kazanır.

- Zararlı Yazılımlar

Zararlı yazılımlar, bilgisayarların ve mobil cihazları işlevselliğini bozmak, kullanıcılar hakkında önemli bilgileri toplamak ve özel sistemlere erişim sağlama amacıyla kullanılan yazılımdır. Zararlı yazılımlar daha çok casusluk işlerinde kullanılmaktadır. Bilgisayar virüsü, Truva atı, solucan, fidye virüsü ve casus yazılım gibi türleri bulunmaktadır.

BÖLÜM 3

YAZILIM TANIMLI AĞLAR ve MİNİNET

3.1 Yazılım Tanımlı Ağlar

Sahba (2018) göre yazılım tanımlı ağlar, ağ alanındaki en son devrimdir. Mevcut donanım altyapısını kullanarak sanal ağlar kurmak ve onu kısa sürede ve düşük maliyetle esnek bir ağ tanımlamak için programlama kilit noktadır. Yazılım tanımlı ağ, programlanabilir olma ve esnek ağ modeli sunmaktadır. Öte yandan, geleneksel bilgisayar ağları satıcı bazlıdır ve ağ yapılandırmasında herhangi bir değişiklik yapmak oldukça zor ve maliyetlidir (Dhaya, Maharaj, Sowmya & Kanthavel, 2017). YTA özellikli bir altyapı ağ kontrolünü yönlendirme işleminden ayırması ve doğrudan programlanabilmesi nedeniyle YTA için yeni bir ağ paradigması, yeni bir mimari ve yeni ilkeler bütünü olduğuna dikkat çekmişlerdir (King, Rotsos, Aguado, Georgalas & Lopez, 2016).

İlk olarak Stanford Üniversitesi'nde OpenFlow çalışmalarının ardından ortaya atılan yazılım tanımlı ağlar terimi, veri düzlemi ve kontrol düzleminin birbirinden ayrılarak oluşturulan ağ mimarisini ifade etmektedir (McKeown Anderson, Balakrishnan, Parulkar, Peterson, Rexford & Turner, 2008). YTA'ların gelişimi programlanabilir ağlar üzerine olmuştur. Aktif ağlar (Tennenhouse, Smith, Sincoskie, Wetherall & Minden, 1997), Programlanabilir ATM ağları (Lazar, Lim & Marconcini, 1996), Network Control Point (NCP) (Sheinbein ve Weber, 1982) ve Routing Control Platform (RCP) (Caesar,

Cladwell, Feamster, Rexford, Shaikh & van der Merwe, 2005) programlanabilen ağların en bilinenleridir (Alparslan, 2016). Programlanabilir ağlar, ağlarda yük dengeleme ve ağ güvenliği vb. konularda 1990'lı yıllarda önerilmiştir ve ağ cihazlarından anahtarlayıcı kullanılmıştır. Fakat ağ cihazları üreten firmalar, kendi kurallarına ya da müsaade ettikleri kadarına açık olan programlanabilir ağlar farklı üreticilerden olan ağ cihazlarının aynı yapılandırmayı sağlayamamasından dolayı kısıtlı başarıya ulaşmıştır (Alparslan, 2016).

YTA teknolojisi, ağ performansını arttırmak ve geleneksel bilgisayar ağı yönetimine göre daha dinamik, programlanabilir olması sayesinde etkin bir ağ yapılandırması sağlayan ağ yönetimi yaklaşımıdır. Geleneksel bilgisayar ağ mimarisinin daha merkeziyetçi ve karmaşık olduğundan dolayı, mevcut ağların esneklik ve sorun giderme gibi durumlarda eksiklikleri bulunmaktadır (De Oliveira, Schweitzer, Shinoda & Prete, 2014). YTA, ağ paketlerinin (veri düzlemi) yönlendirme işlemini iletim işleminden (kontrol düzlemi) ayırarak ağ zekasını bir ağ bileşeninde merkezileştirmeye çalışmaktadır. Kontrol düzlemi, tüm zekanın dahil olduğu YTA ağının beyni olarak kabul edilen bir veya daha fazla kontrolcüden oluşmaktadır (Sdxcentral, 2014).

Ağ sanallaştırması YTA'nın gelişimi ile araştırma topluluğunda büyük önem kazanmıştır. Programlanabilir ağlarda olduğu gibi, ağ sanallaştırması 1990'lı yıllarda ortaya çıkan bir konuydu. Tempest Proje (Van der Merwe, Rooney, Leslie & Crosby, 1998)'si bizlere, ağ sanallaştırmasını tanıtan ilk projelerden biridir. Bu projenin ana fikri, birden fazla ATM ağının tek bir ATM anahtarlayıcısı üzerinden aynı fiziksel kaynakları paylaşmasını sağlamaktı. Benzer şekilde Mbone (Macedonia ve Brutzman, 1994) ağlar üzerinde sanal ağlar oluşturmayı hedefleyen girişimlerden biriydi. Bu projeyi daha sonra PlanetLab (Chun, Culler, Roscoe, Bavier, Peterson, Wawrzoniak & Bowman, 2003), GENI (Peterson, Anderson, Blumenthal, Casey, Clark, Estrin & Shenker, 2006) ve VINI (Bavier, Feamster, Huang, Peterson & Rexford, 2006) takip etmiştir (Kreutz vd., 2015).

OpenFlow çalışmaları ile birlikte, geleneksel bilgisayar ağlarının kontrol katmanı ve veri katmanının birbirinden ayırma fikrini SANE (Casado, Garfinkel, Akella, Freedman, Boneh, McKeown & Shenker, 2006), Ethane (Casado, Freedman, Pettit, McKeown & Shenker, 2007) ve NOX (Gude, Koponen, Pettit, Pfaff, Casado, McKeown & Shenker, 2008) gibi çalışmalar öne sürmüştür. Bu çalışmalar, yönlendirme cihazlarında önemli

değişiklikler gerektirmediğinden onları yalnızca araştırma topluluğu için değil, ağ endüstrisi için daha da cazip hale getirmektedir (Kreutz vd., 2015).

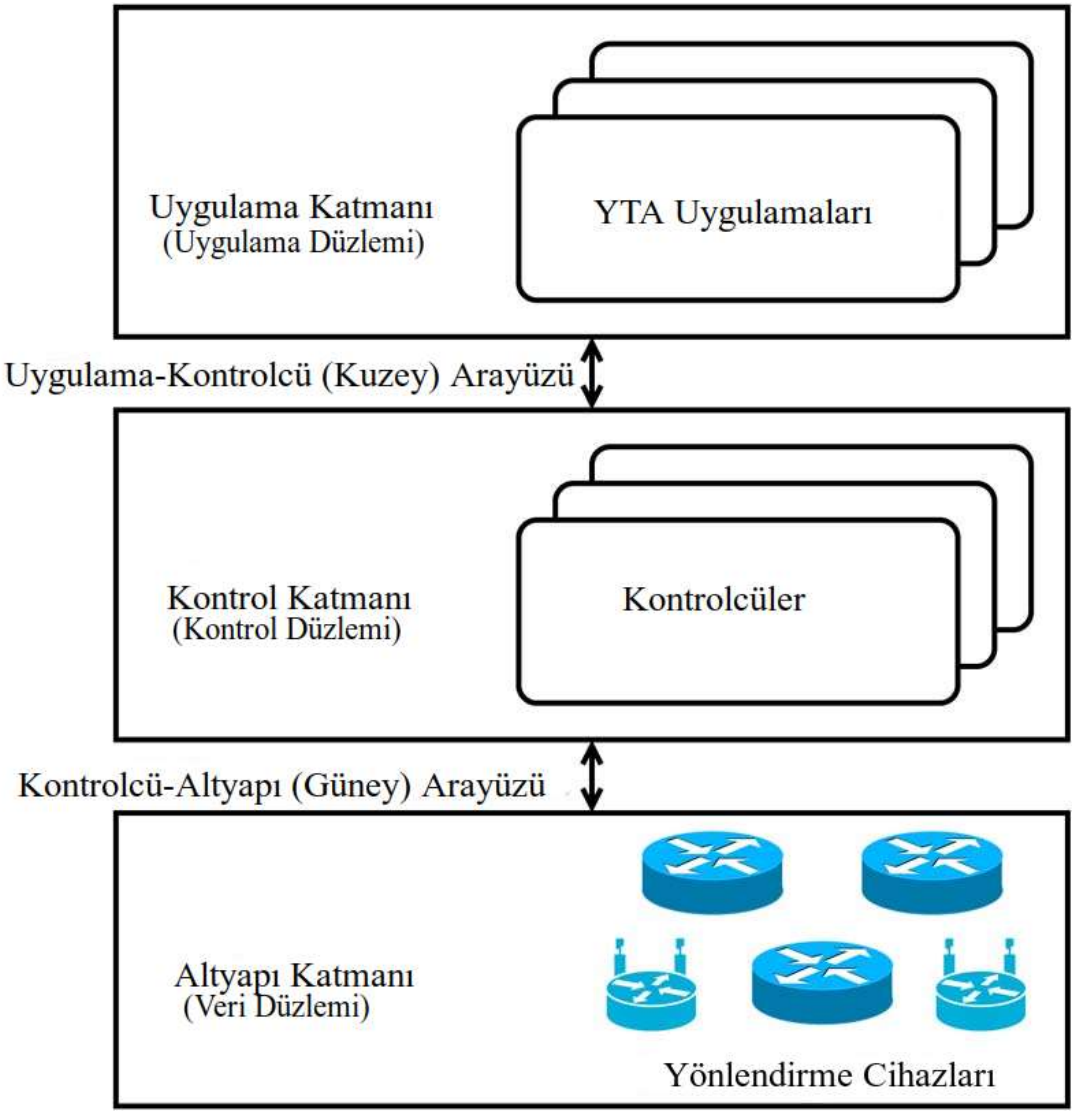
3.1.1. OpenFlow Protokolü

Nick Mckeown ve arkadaşları (2008), OpenFlow kavramını ortaya atmışlardır. OpenFlow ağ anahtarlayıcılarının ve yönlendiricilerinin ağ üzerinden yönlendirme düzlemine erişim sağlayan bir iletişim protokolüdür. Yazılım tanımlı ağ standartların ilki kabul edilmektedir. OpenFlow, YTA kontrolcüsünün hem fiziksel hem de sanal anahtarlayıcılar ve yönlendiriciler gibi ağ cihazlarının yönlendirme düzlemi ile doğrudan etkileşime girmesini sağlayan YTA ortamlarında tanımlanmıştır. 2011 yılında OpenFlow üzerine çalışmalar yapmak amacıyla Open Networking Foundation (ONF) kurulmuştur.

Güncel olarak kullanılan YTA kontrolcüleri OpenFlow protokolüne dayanmaktadır. YTA kontrolcüsü, ağ için bir tür işletim sistemi görevi görmektedir. Uygulamalar ve cihazlar arasındaki tüm iletişim kontrol cihazları üzerinden geçmektedir. OpenFlow protokolü, kontrolcü yazılımı ağ cihazlarına bağlamaktadır. Böylece sunucu yazılımı, anahtarlayıcılara paketlerin nereye gönderileceğini söylemektedir. Kontrolcü, ağ cihazlarını yapılandırmak ve uygulama trafiği için en iyi yolu seçmek için OpenFlow protokolünü kullanmaktadır. Ağ kontrol düzlemi donanım cihazlarının yazılımı yerine yazılımda uygulandığından, ağ trafiği daha dinamik ve daha ayrıntılı bir düzeyde yönetilebilir.

3.1.2. Yazılım Tanımlı Ağ Mimarisi

ONF, yazılım tanımlı ağ mimarisi için 3 katmanlı mimari önermiştir. Bu katmanlar sırasıyla: altyapı katmanı kontrol katmanı ve uygulama katmanıdır (Opennetworking, 2014). Şekil 3.1'de bu katmanlar gösterilmiştir. Bu katmanlar aynı zamanda sırasıyla veri düzlemi, kontrol düzlemi ve uygulama düzlemi olarak da adlandırılmaktadır.



Şekil 3.1. ONF'nin Önerdiği YTA Mimarisi

Diego Kreutz ve arkadaşları, ONF'nin önermiş olduğu 3 katmanlı mimariyi geliştirerek her bir katmanın kendine özgü fonksiyonlarının olduğu 8 katmanlı bir yapı önermiştir (Kreutz vd., 2015). Bu katmanlardan bazıları her YTA dağıtımında bulunurken, diğer katmanlar sadece belirli dağıtımlarda bulunabilmektedir. Bu 8 katmanlı yapı Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1. ONF ve Diego Kreutz Önerdiği Mimarilerin Karşılaştırmalı Gösterimi

ONF'nin Önerdiği YTA Mimarisi	Diego Kreutz'un Önerdiği YTA Mimarisi
Ağ Düzlemi	Ağ Uygulamaları
	Programlama Dilleri
	Dil Tabanlı Sanallaştırma
Kontrol Düzlemi	Kuzey Arayüzü
	Ağ İşletim Sistemi
	Ağ Yüksek Yöneticisi
Veri Düzlemi	Güney Arayüzü
	Ağ Altyapısı

ONF'nin önermiş olduğu 3 katmanlı mimaride bulunan veri düzlemi, kontrol düzlemi, uygulama düzlemi ve katmanlar arasında haberleşmeyi sağlayan kuzey ve güney arayüzlerinin özellikleri şöyledir:

3.1.2.1. Veri Düzlemi

Veri düzlemi sanallaştırma, bağlantı, güvenlik, kullanılabilirlik ve kaliteyi sağlamak için gerekli destekleyici kaynaklarla birlikte doğrudan ağ trafiğiyle ilgilenen kaynakları birleştirmektedir. Yazılım tarafından tanımlanan ağ iletişimi, trafik iletme ve servis kalitesi, filtreleme, izleme ve trafik işleme gibi görevlerden sorumludur. Kontrol düzleminde verilen iletme kararlarını uygular.

3.1.2.2. Kontrol Düzlemi

Kontrol düzlemi, veri paketlerinin nereye gönderileceğine karar verir (Searchnetworking, 2018). Kontrol düzlemi işlevleri, sistem yapılandırmasını, yönetimini ve yönlendirme tablosu bilgilerinin değişimini içerir. Rota kontrolcüsü, topoloji bilgilerini diğer yönlendiriciler ile iletir ve yönlendirme protokollerine bağlı kalarak yönlendirme tablosunu oluşturur. Kontrol düzlemi paketleri, yönlendirme tablosunun güncellenmesi amacıyla yönlendirici tarafından işlenir. Kontrol fonksiyonları gelen her bir pakette yerine getirilmediğinden, katı bir hız kısıtlaması yoktur ve zaman açısından daha az kritiktir.

3.1.2.3. Uygulama Düzlemi

Uygulama düzlemi, ağ kontrolü ve işlem mantığını uygulamak için kuzey arayüzü tarafından sunulan işlevleri kullanan uygulama kümesidir. Bu düzlem, yönlendirme, güvenlik duvarları, yük dengeleyici, izleme ve benzeri gibi uygulamaları içerir. Temel olarak, bir yönetim uygulaması, nihayetinde, yönlendirme cihazlarının davranışını programlayan güneye özgü talimatlara çevrilen politikaları tanımlar.

3.1.2.4. Kuzey ve Güney Arayüzleri

- Güney arayüzü

İletme cihazlarının komut seti, güney arayüzü ile tanımlanır. Ayrıca, güney arayüzü ayrıca iletme cihazları ve kontrol düzlemi elemanları arasındaki iletişim protokolünü tanımlamaktadır. Bu protokol, kontrol ve veri düzlemi elemanlarının etkileşim şeklini biçimlendirir. OpenFlow burada çalışır. OpenFlow dışında Simple Network Management Protocol (SNMP), File Transfer Protocol (FTP), Telnet ve Secure Shell (SSH) gibi protokoller burada çalışır.

- Kuzey arayüzü

Kuzey arayüzü, bir ağın belirli bir bileşeninin daha yüksek seviyeli bir bileşen ile iletişim kurmasına izin veren bir arayüzdür. Bir kurumsal veri merkezinde, kuzey arayüzünün işlevleri arasında otomasyon ve düzenleme için yönetim çözümleri ve işlem yapılabilir verilerin sistemler arasında paylaşılması bulunur. REST API kullanır.

3.1.3. Yazılım Tanımlı Ağ Kontrolcüleri

Yazılım tanımlı ağ kontrolcüleri, YTA mimarisinde kontrol katmanında yer almaktadır. YTA kontrolcüsü, gelişmiş ağ yönetimi ve uygulama performansı için akış kontrolünü yöneten, YTA mimarisindeki bir uygulamadır. YTA kontrolcüsü platformu tipik olarak bir sunucu üzerinde çalışır ve anahtarlayıcıların paketleri nereye gönderileceğini anlatmak için protokoller kullanır (Searchnetworking, 2013).

YTA kontrolcüleri, veri trafiğini bir şebeke operatörünün uyguladığı yönlendirme politikalarına göre doğrudan yönlendirmekte ve böylece bireysel ağ cihazları için elle yapılan ayarlamaları en aza indirmektedir. Kontrol düzlemini ağ donanımından çıkartarak

bunun yerine yazılım olarak çalıştırmak, merkezi kontrol birimi otomatik ağ yönetimini kolaylaştırır, iş uygulamalarını entegre etmeyi ve yönetmeyi kolaylaştırır. YTA kontrolcüsü, ağ için bir tür işletim sistemi görevi görmektedir.

Kontrolcü, yazılım tanımlı bir ağın çekirdeğidir. Ağın bir ucundaki ağ cihazları ve diğer ucundaki uygulamalar arasında bulunur. Uygulamalar ve ağ cihazları arasındaki tüm iletişim denetleyiciden geçmektedir.

Kontrolcüler uygulamalarla güvenlik duvarı ve yük dengeleyicileri gibi kuzey arayüzü üzerinden iletişim kurmaktadır. ONF 2013 yılında kuzey arayüz API'leri ve onların geliştirilmesi için bir grup kurmuştur (Opennetworking, 2013).

Kontrolcü, güney arayüzünde yer alan OpenFlow gibi protokolleri kullanarak, ağda bulunan her bir cihazla iletişim kurmaktadır. Güney arayüz protokolleri, kontrolcünün uygulama trafiği için en uygun ağ yolunu seçmesi ve ağ cihazlarının yapılandırılmasına olanak sağlamaktadır.

Kullanıldığı alanlara göre birçok açık kaynak kodlu YTA kontrolcüsü programlanmıştır. NOX (Gude vd., 2008), POX ve Floodlight kullanım alanı en geniş olan YTA kontrolcüleridir (Alparslan, 2016). Çizelge 3.2’de açık kaynak kodlu YTA kontrolcüleri hakkında bilgiler verilmiştir.

Çizelge 3.2. YTA Kontrolcü Özellikleri (Salman, Elhajj, Kayssi ve Chehab, 2016)

	Programlama Dili	**GUI**	**Dokümantasyon**	**Platform Desteği**	**Güney Arayüzü**	**Kuzey Arayüzü**
ONOS	Java	Web Tabanlı	İyi	Linux, MAC OS ve Windows	OF 1.0, 1.3 NETCONF	REST API
Openday-light	Java	Web Tabanlı	Çok İyi	Linux, MAC OS ve Windows	OF1.0, 1.3, 1.4, NETCONF/ YANG, OVSDB, PCEP, SNMP	REST API
NOX	C ++	Python +QT4	Zayıf	Çoğunlukla Linux destekli	OF 1.0	REST API
POX	Python	Python +QT4	Zayıf	Linux, MAC OS ve Windows	OF 1.0	REST API
RYU	Pyhon	-	Orta	Çoğunlukla Linux destekli	OF 1.0, 1.2, 1.3, 1.4, NETCONF, OF-CONFIG	Güney arayüzü için REST
Floodlight	Java	Web	İyi	Linux, MAC OS ve Windows	OF 1.0, 1.3	REST API
Maestro	Java	-	Orta	Linux, MAC OS ve Windows	OF 1.0	REST API
Beacon	Java	Web Tabanlı	Orta	Linux, MAC OS ve Windows	OF 1.0	REST API
Iris	Java	Web Tabanlı	Orta	Linux, MAC OS ve Windows	OF 1.0, 1.3, OVSDB	REST API

3.1.4. Yazılım Tanımlı Ağlarda Güvenlik

Yazılım tanımlı ağlar, ağ yöneticilerine basitlik, programlanabilirlik ve esneklik açısından kolaylık sağlamaktadır. Bu kolaylıklar bilim dünyası ve endüstriyel şirketler tarafından büyük ilgi ile karşılamıştır. Yazılım tanımlı ağların güvenliği üzerine araştırmalar yapılmasına rağmen hala yeteri kadar ilgi gösterilmemiştir.

Yazılım tanımlı ağlar, tasarımına eklediği bazı özellikler sayesinde geleneksel bilgisayar ağlarının güvenlik sorunlarından bazılarını çözmektedir. Fakat kontrol ve veri katmanlarının birbirinden ayrılmasından dolayı yeni güvenlik sorunları ortaya çıkmıştır. Bu güvenlik sorunları yazılım tanımlı ağlarda, ağ trafiğinin ve verilerin gizliliği, özgünlüğü, bütünlüğü ve tutarlılığını etkilemektedir.

Yazılım tanımlı ağlarda saldırı ve güvenlik açıklarını yetkisiz erişim, veri sızıntısı, veri değişikliği, kötü amaçlı/tehlike uygulamalar, servis reddi, yapılandırma sorunları ve yazılım tanımlı ağların sistem seviyesi güvenliği olduğunu belirtmişlerdir (Sezer vd., 2013).

Güvenlik açıklarını katmanları teker teker inceleyip her katmanın güvenlik açıklarını ortaya koymuşlardır. Bu güvenlik açıklıklarını uygulama katmanında kimlik doğrulama ve yetkilendirme, kontrol düzleminde uygulamalardan kaynaklanan tehditler, ölçeklenebilirlik nedeniyle tehditler, DoS saldırıları ve dağıtık kontrol düzlemindeki zorluklar ve veri düzleminde bulunan zorluklar olarak ortaya koymuşlardır (Ahmad, Namal, Ylianttila & Gurtov, 2015).

Diego Kreutz ve diğerleri (2015), yazılım tanımlı ağ mimarisinde 7 tane tehdit vektörü olduğunu belirtmişlerdir. Bu tehdit vektörlerinden dördü geleneksel bilgisayar ağlarında da bulunurken, üçü yazılım tanımlı ağlara özeldir. Bu 3 tehdit vektörü yazılım tanımlı ağlar için en tehlikeli olanıdır. Tehdit vektörleri Çizelge 3.3'de gösterilmektedir.

Çizelge 3.3. Tehdit Vektörlerinin Açıklaması

Tehdit Vektörü	YTA'ya Özel mi?	Sonuçları
Vektör 1	Hayır	DoS saldırılarına açık kapı
Vektör 2	Hayır	Olası saldırı enflasyonu
Vektör 3	Evet	Mantıksal merkezileştirilmiş kontrolcülerin sömürülmesi
Vektör 4	Evet	Tüm ağın güvenliğinin tehlikeye atılması
Vektör 5	Evet	Kontrolcüde kötü amaçlı yazılımların kullanılması
Vektör 6	Hayır	Olası saldırı enflasyonu
Vektör 7	Hayır	Hata teşhisinde ve sorun çözümüne olumsuz etki

- **Tehdit Vektörü 1**

Yönlendiricilere ve anahtarlayıcılara saldırmak için ağın içinde sahte trafik oluşturarak kullanılan saldırıdır. OpenFlow anahtarlayıcılarına ve kontrolcün kaynaklarına DoS saldırıları başlatmak için kullanılır.

- **Tehdit Vektörü 2**

Ağda bulunan anahtarlayıcıların güvenlik açıklarına saldırır. Ele geçirilen bir anahtarlayıcı ağ trafiğini kopyalamak, saptırmak, ağda bulunan paketleri yavaşlatmak, düşürmek ve sahte ağ trafiği oluşturarak ağ bulunan cihazları aşırı yüklemek gibi sorunlara sebep olabilir.

- **Tehdit Vektörü 3**

Veri hırsızlığı veya DoS saldırıları oluşturmak için kontrol düzlemine yapılan saldırıdır. Transport Layer Security (TLS) / Secure Socket Layer (SSL) güvenli iletişimi

garanti etmemektedir. TLS / SSL kontrolcüler ve anahtarlayıcılar arasında güven sağlamak için yeterli değildir. Bir saldırgan kontrol düzlemine eriştiğinde sanal kara delik oluşturarak ağ trafiğinde veri sızıntısı ve DoS saldırısı yapabilir.

- **Tehdit Vektörü 4**

Yazılım tanımlı ağlara yönelik en tehlikeli saldırı vektörüdür. Hatalı ya da kötü amaçlı bir kontrolcü tüm ağın güvenliğini tehlikeye atabilir. Kontrolcünün davranışlarını değiştirebildiğinden kontrolcü ağda bulunan zararlı yazılımlara engel olamayabilir.

- **Tehdit Vektörü 5**

Kontrolcü ve uygulamalar arasında güvenilir ilişki kuramaması nedeniyle tehdit vektörü 3'e benzemektedir. Kontrolcüde kötü amaçlı yazılımların kullanılmasına sebep olur.

- **Tehdit Vektörü 6**

Geleneksel bilgisayar ağlarında da görülen bu saldırı, yönetici istasyonlarındaki güvenlik açıklarından faydalanır. Bu saldırının yazılım tanımlı ağlarda tehdit düzeyi oldukça yüksektir.

- **Tehdit Vektörü 7**

Tespit edilen sorunun nedenini anlamaya ve hızlı bir şekilde çözmesine engel olmak için kullanılan saldırıdır. Ağın güvenilirliğini tehlikeye atmaktadır.

3.2. Mininet

Mininet, sanal ana bilgisayar, anahtarlayıcı, kontrolcü ve bağlantı ağı oluşturan bir ağ emülatörüdür. Mininet hostları standart Linux ağ yazılımını çalıştırır ve anahtarlayıcıları, oldukça esnek ve özel yönlendirme ve yazılım tanımlı ağlar için OpenFlow'u desteklemektedir. Mininet tarafından oluşturulan OpenFlow anahtarlayıcıları, ağda bulunan cihazlara paket iletimini yapmaktadır (De Oliveira vd, 2014).

Mininet, araştırma, geliştirme, öğrenme, prototip oluşturma, test etme, hata ayıklama ve bir dizüstü bilgisayarda tam bir deneysel ağa sahip olmanın faydası olabilecek diğer işleri destekler.

Mininet:

- OpenFlow uygulamaları geliştirmek için basit ve ucuz bir test ağı sağlar.
- Birden fazla eş zamanlı geliştiricinin aynı topoloji üzerinde bağımsız olarak çalışmasını sağlar.
- Fiziksel bir ağ bağlanmaya gerek kalmadan karmaşık topoloji testi sağlar.
- Ağ sınaması, hata ayıklamak veya çalıştırmak için topoloji ve OpenFlow uyumlu bir Command Line Interface (CLI) içerir.
- İsteğe bağlı özel topolojileri destekler.
- Programlama olmadan kullanılabilir.
- Ağ oluşturma ve deneme için basit ve genişletilebilir bir Python API sağlar.

Mininet ağları, standart Unix / Linux ağ uygulamalarının yanı sıra gerçek Linux çekirdeği ve ağ yığını (ağ ad alanlarıyla uyumlu oldukları sürece kullanılabilir olabilecek tüm çekirdek uzantıları dahil) dahil olmak üzere gerçek kod çalıştırmaktadır (Mininet, 2018). Bu nedenle, bir OpenFlow denetleyicisi, değiştirilmiş anahtarlayıcı veya ana bilgisayar için Mininet'te geliştirilen ve test edilen kod, gerçek dünya testi, performans değerlendirmesi ve dağıtım için minimum değişikliklerle gerçek bir sisteme geçebilir. Önemli olan, bu, Mininet'te çalışan bir tasarımın, genellikle satır oranı paket iletme için doğrudan donanım anahtarlayıcı geçebileceği anlamına gelir.

Mininet, host çalıştırmak ve tek bir işletim sistemi çekirdeğini açmak için işlem tabanlı sanallaştırmayı kullanır. 2.2.26 sürümünden bu yana Linux, ayrı ağ arayüzleri, yönlendirme tabloları ve ARP tabloları ile bireysel işlemler sağlayan hafif bir sanallaştırma özelliği olan ağ ad alanlarını desteklemektedir. Mininet, çekirdek veya kullanıcı alanı OpenFlow anahtarlayıcıları, anahtarlayıcıları kontrol etmek için

kontrolcüler ve benzetilmiş ağ üzerinden iletişim kurmak için hostlar oluşturabilir. Mininet, anahtarlayıcıları ve hostları sanal ethernet çiftleri kullanarak bağlar (Mininet, 2018).

3.2.1 Mininet Topolojileri

Mininet minimal, single, reversed, linear ve tree gibi varsayılan topolojiler içermektedir. Varsayılan topolojilerin dışında Python programalama dili kullanılarak isteğe bağlı özel topolojiler de oluşturabilir. Arayüzlerin, hostların ve anahtarlayıcıların adlandırılmasını anlamak, Mininet kullanırken gereklidir. Anahtarlayıcılar s1'den sN'e kadar adlandırılır. Hostlar h1'den hN'e kadar adlandırılır. Her bir hostun ethernet adı "0" ile başlar yani h1 hostunun ethernet arayüzü h1-eth0 olarak adlandırılır. İlk anahtarlayıcı "s1" olarak adlandırılır ve hosta bağlanan ilk portu s1-eth1 diye adlandırılır (Mininet, 2018).

- **Minimal Topoloji**

Minimal topoloji çok basit bir topolojidir. 1 tane anahtarlayıcı ve 2 tane hosttan oluşur. Terminale **#mn --topo minimal** yazarak oluşturulur (Kaur, Singh ve Ghumman, 2014).

- **Single Topoloji**

Bir anahtarlayıcı ve k tane host ile oluşturulan topolojidir. Terminale **#mn --topo single,k** yazılarak oluşturulur (Kaur vd., 2014).

- **Reversed Topoloji**

Single topolojiye çok benzemektedir. Tek farkı bağlantı düzeni terstir. Terminale **#mn --topo reversed,k** yazılarak oluşturulur (Kaur vd., 2014).

- **Linear Topoloji**

Linear topoloji k tane anahtarlayıcı ve k tane hosttan oluşan topolojidir. Her bir anahtarlayıcı kendi arasında ve anahtarlayıcı ile host arasındaki bağlantı kurulur. Terminale **#mn --topo linear,k** yazılarak oluşturulur (Kaur vd., 2014).

- **Tree Topoloji**

Tree topoloji k seviyede olur ve her bir anahtarlayıcıya en az 2 tane host bağlı olmaktadır. Terminale **#mn --topo tree,k** yazılarak topoloji oluşturulmuş olur (Kaur vd., 2014).

- **Özel Topoloji**

Python programlama dilini kullanarak Mininet'te kolaylıkla özel topoloji oluşturulabilir (Kaur vd., 2014). Şekil 3.2'de 2 host ve 2 anahtarlayıcıdan oluşan örnek özel topoloji kodu gösterilmektedir.

```
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Şekil 3.2. Python Kullanarak Oluşturulan Özel Topoloji

3.2.2. Mininet Ağı Oluşturmak İçin Kullanılan Komutlar

Mininet kullanılarak ağ oluşturmak ve özelleştirmek için kullanılan komutlar vardır. Bu komutlar kullanılarak ağ büyüklükte ve özellikte oluşturulabilir.

- **mn:** Mininet CLI'ya girmek için kullanılır.
- **-h, --help:** yardım mesajlarını gösterir.
- **--switch:** oluşturulan ağda kullanılan anahtarlayıcının türünü belirler.
- **--controller:** kontrolcünün türünü belirler.
- **--topo:** oluşturulan ağın topoloji türünü belirler.

3.2.3. Mininet CLI'da Kullanılan Basit Komutlar

Ağı oluşturduktan sonra, ağ içinde çeşitli ağın kontrollerini ve ağın durumu hakkında bilgi sahibi olmayı sağlayan komutlardır. Şekil 3.3'te bu komutların neler olduğu gösterilmektedir.

```
mininet> help

Documented commands (type help <topic>):
========================================
EOF    exit   intfs     link   noecho       pingpair      py    source  xterm
dpctl  gterm  iperf     net    pingall      pingpairfull  quit  time
dump   help   iperfudp  nodes  pingallfull  px            sh    x
```

Şekil 3.3. Mininet CLI'da Kullanılan Basit Komutlar

- **help:** mevcut komutların listesini görmek için kullanılır.
- **nodes:** mevcut düğümlerin listesini görmek için kullanılır.
- **dump:** benzetim ağındaki düğümler, anahtarlayıcılar ve kontrolcüler hakkında bilgileri listeler.
- **list:** hangi anahtarlayıcıların ve hostların bağlı olduğunu gösterir.
- **net:** linke benzer, benzetimi ağındaki ağ elemanlarının birbirlerine nasıl bağlandığını gösterir.

- **ping:** belirli hostlar arasındaki bağlantıyı gösterir.
- **pingall:** tüm hostların arasındaki bağlantıyı gösterir ve bize hangi hostların birbirine bağlı olduğunu gösterir.
- **pingallfull:** hostların nasıl bağlandıkları hakkında daha fazla ayrıntı verir. İki host arasında milisaniye cinsinden minimum, ortalama ve maksimum süreyi gösterir.
- **iperf:** genellikle TCP bağlantılarında kullanılır. Hostlar arasında bant genişliğini test etmek için kullanılır.
- **eof :** mevcut topolojiden çıkmak için kullanılır.
- **exit:** eofa benzer, ayrıca topolojiden çıkmak için kullanılır.
- **iperfudp:** iperfe benzer, TCP bağlantısı yerine UDP bağlantısı kullanır.
- **ifconfig:** sanal hostların IP adreslerini öğrenmek için kullanılır.
- **xterm:** düğüme bağlı bir pencere açmak için kullanılır.
- **intfs:** bağlı arayüzleri gösterir.
- **dpctl:** anahtarlayıcı tablosundaki akışı göstermek için kullanılır.

BÖLÜM 4

MATERYAL METOD

Bu bölümde yazılım tanımlı ağlar ile geleneksel bilgisayar ağları arasında güvenlik karşılaştırmalarının benzetim çalışmalarını yapmak için test ortamlarının nasıl oluştuğu, yapılacak siber saldırılar ve saldırıların nasıl yapılacağı anlatılmıştır. Yazılım tanımlı ağların test ortamını oluşturmak için Ubuntu 14.04 LTS işletim sistemi kullanılmıştır. Yazılımı tanımlı ağ kontrolcüsü olarak Floodlight kontrolcüsünün v1.1 sürümü kullanılmıştır. Floodlight v1.1'i kullanmak için sanal makineye Java JDK 1.7, Ant derleyicisi, Maven, Python ve Eclipse yüklenmiştir. Topolojiyi oluşturmak için Mininet emülatörü kullanılmıştır. Yazılım tanımlı ağlarda yapılacak olan siber saldırılar topoloji içinde yer alan hostlar ile yapılmıştır. Geleneksel bilgisayar ağlarında siber saldırıları yapmak için Kali Linux işletim sistemi kullanılmıştır. Her iki işletim sistemi de VirtualBox üzerinde lokal olarak çalıştırılmıştır.

4.1. Test Ortamlarının Hazırlanması

Test ortamını oluşturmaya başlamak için en önce, VirtualBox uygulamasını bilgisayara yüklenmelidir. Daha sonra Ubuntu 14.04 LTS sanal makinesi VirtualBox'a kurulmalıdır. Floodlight YTA kontrolcüsünün çalışması için sanal makineye yüklenmesi

gereken bazı uygulamalar vardır. Bunlar: Java JDK, derlemek için Ant veya Maven, Python geliştirme paketi ve Eclipse IDE.

Çalışmada Floodlight'ın v1.1 sürümü kullanılmıştır. Floodlight v1.1'in çalışması için gerekli olan uygulamaları indirmek için komut satırına **sudo apt-get install build-essential openjdk-7-jdk ant maven python-dev eclipse** yazılır. Daha sonra Floodlight'ı GitHub üzerinden komut satırına **git clone git://github.com/floodlight/floodlight.git -b v1.1** yazılarak Floodlight kontrolcüsü indirilir (Floodlight, 2012).

Ubuntu sanal makinesine Mininet emülatörü kurmak için önce terminale **git clone git://github.com/mininet/mininet** yazılarak Mininet'in dosyalarını indirilir. Daha sonra terminal ekranına **mininet/util/install.sh** yazarak Mininet'in bilgisayara kurulum işlemi başlatılır (Mininet, 2018).

Geleneksel bilgisayar ağlarına yapılacak saldırılar Kali Linux sanal makinesi üzerinde yapılmıştır. Kali Linux kendi sitesinden VirtualBox ile uyumlu olan ova uzantılı imaj dosyası indirilmiştir. Daha sonra gerekli ayarlamalar yapılıp VirtualBox'ta Kali Linux sanal makinesi çalışır hale getirilmiştir.

Kali Linux sızma testi ve etik bilgisayar korsanlığı için kullanılan bir Debian dağıtımıdır (Kali, 2013). Yapılacak saldırılarda kullanılan her uygulama Kali Linux işletim sisteminin içinde hazır olarak gelmektedir. Aynı saldırıları yazılım tanımlı ağlarda yapmak için gerekli uygulamalar Ubuntu sanal makinesine indirilmiştir.

Hem geleneksel bilgisayar ağlarının hem de YTA'ların güvenlik karşılaştırması için Distributed Denial of Service saldırısı (DDoS) ve man-in-the-middle saldırıları yapılmıştır. DDoS saldırısı için hping3 ve man-in-the-middle saldırısı için de dsniff uygulaması kullanılmıştır.

Dsniff programı man-in-the-middle saldırısını yaparken kullanılan programdır (Monkey, 2000). Dsniff ileri düzey araçların bulunduğu açık kaynak kodlu çoğu Linux dağıtımında bulunan ağ izleme ekrana ağdaki şifreleri yazdırmak için ağ güvenliğini

sağlamak ve daha birçok şey yapabilmek için kullanılan bir programdır. ARP soof, DNS spoof gibi birçok araç içeren güvenlik programıdır.

- **sudo apt-get update:** apt-get update, sanal makinede bulunan paketlerin en son sürümüne güncellenmesini sağlar.

- **sudo apt-get install dsniff:** paketler güncelleştirildikten sonra, komut satırına apt-get install dsniff yazılarak dsniff programı sanal makineye yüklenmiştir.

Ping programının gelişmiş versiyonudur. Hpin3 tıpkı ping programının yaptığı gibi hedef IP adresine paketler göndermektedir. Hping3 uygulaması istenilen kriterlere göre paket oluşturmayı sağlamaktadır. Hping3 ağ testleri, güvenlik duvarı testi, DoS ve DDoS saldırıları için kullanılmaktadır (Hping, 2006).

- **sudo apt-get update:** apt-get update, sanal makinede bulunan paketlerin en son sürümüne güncellenmesini sağlar.

- **sudo apt-get install hping3:** paketleri güncelleştirdikten sonra, komut satırına apt-get install hping3 yazılara hping3 programı sanal makineye yüklemiş olur.

Ubuntu sanal makinemize gerekli uygulamalar yüklendikten sonra yazılım tanımlı ağlar oluşturulmuştur. Yazılım tanımlı ağların oluşturulması için Floodlight kontrolcüsünün çalışır hale getirilmesi gerekmektedir. Ubuntu sanal makinesinde terminal açılıp buradan indirilmiş olunan Floodlight klasörünün içine girilir. Terminalde **sudo ant** komutunu çalıştırılarak Floodlight kontrolcüsünü derlenmiş olur. Derleme işlemi bittikten sonra **java -jar ./target/floodlight.jar** komutu çalıştırılır. Bu işlemlerin ardından Floodlight kontrolcüsü kullanıma hazır hale gelmektedir (Floodlight, 2012).

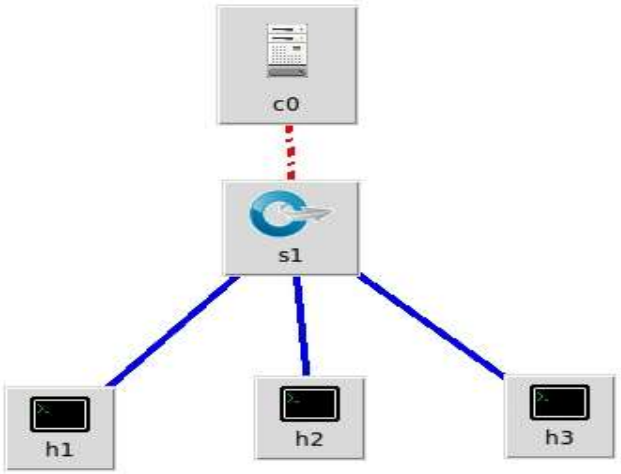
Benzetim çalışmasında kullanılacak topolojiyi oluşturmak için Mininet'in varsayılan topolojilerden yararlanılmaktadır. Ağda 1 anahtarlayıcı ve 3 host bulunmaktadır. Ağın varsayılan Mininet topoloji türü olarak da tree topolojisi kullanılmıştır. Ağın topolojisi Şekil 4.1'de gösterilmektedir. Topolojiyi oluşturmak için terminale **sudo mn --controller=remote,ip=127.0.0.1,port6653 --topo=tree,1,3** yazılır.

- **--controller=remote:** remote parametresi, kontrol ağı üzerinde herhangi bir yerde çalışan, ancak manuel olarak veya Mininet'in doğrudan kontrolü dışındaki başka bir mekanizma tarafından başlatılması ve kapatılması gereken bir kontrolcü için vekil olarak davranır.

- **ip=127.0.0.1:** kontrolcüye Mininet üzerinden atanan IP adresidir.

- **port=6653:** Floodlight kontrolcüsü çalışmaya başladığında 6653 portunu dinlemeye başlar. Oluşturulan topolojiyi kontrolcüye bağlamak için port adresi olarak 6653'ü atanır.

- **--topo=tree,1,3:** ağ topolojisinin varsayılan Mininet topoloji türü olarak tree topolojisi, derinliğinin 1 olduğu ve anahtarlayıcıya bağlı olan host sayısının 3 olarak atandığı belirtilmektedir.



Şekil 4.1. Mininet Yardımıyla Oluşturulan Ağın Topolojisi

4.2. Benzetim Çalışmalarında Kullanılan Saldırılar

Geleneksel bilgisayar ağları ve YTA'ların güvenlik karşılaştırmasını yapmak için her iki ağ türüne de aynı tür siber saldırı yapılmıştır. Bu saldırılar, ağda bulunan hostlar

arasında gerçekleşmiştir. Çalışmada kullanılan saldırılar şunlardır: IP Spoofing, SYN Flooding, RST/FIN Flood, SYN/ACK Flood, UDP Flood, DDoS ve ARP Poisoning.

4.2.1. IP Spoofing

Spoofing, bir kullanıcının, cihazın veya müşterinin internette kimliğine bürünmedir. IP Spoofing, gönderenin kimliğini gizlemek, başka bir bilgisayar sistemini taklit etmek veya bunların her ikisini gerçekleştirmek için değiştirilmiş bir kaynak adresine sahip IP paketlerinin üretilmesidir. Siyah şapkalı bilgisayar korsanları tarafından genellikle hedef cihaz veya çevresindeki altyapıyı hedefleyen DDoS saldırıları gerçekleştirmek için kullanılan bir tekniktir.

Bu saldırıyı yapmak için komut satırını **hping3 -S <hedef_ip> --spoof <rastgele_girilen_ip> --flood** yazılarak saldırı başlatılmış olur.

- **-S:** -S parametresi, saldırı yapılacak IP adresine SYN paketleri gönderileceğini belirtir.

- **--spoof <rastgele_girilen_ip>:** --spoof parametresi saldırıyı yapan makinenin IP adresini gizlemek için kullanılır. Saldırıyı yaparken kullanılan IP adresinin istenildiği gibi girilmesini sağlar.

- **--flood:** --flood parametresi saldırı yapılan hosta paketleri en hızlı biçimde gönderir.

4.2.2.SYN Flooding

SYN Flooding saldırısı, bir sunucunun TCP bağlantı yeteneğinden yararlanan bir saldırıdır. Bir istemci ve bir sunucu üçlü el sıkışması kullanarak bir TCP bağlantısı oluşturmaktadır. Bu saldırı TCP bağlantısı esnasındaki bilinen bir zayıflıktan yararlanmaktadır. SYN Flooding saldırısı sırasında, saldırganın istemcisi hedef sunucuya çok sayıda SYN mesajı gönderir. Sunucu aldığı her SYN için bağlantı tablosunda bir giriş oluşturur ve her birine bir SYN-ACK mesajıyla yanıt verir. Saldırgan ya ACK mesajı

göndermez ya da çoğu kez istemcisinin IP adresini SYN paketlerinde yanlış bildirir, bu sayede hedef sunucunun SYN-ACK yanıtları asla alınmaz. Saldırgan SYN mesajı göndermeye devam ettikçe, hedef sunucunun bağlantı tablosu tümüyle dolar ve sunucu hiçbir bağlantı isteğine yanıt veremez hale gelir. Tüketilen tüm kaynakları sonucunda, hedef sunucu hizmet reddi oluşturur ve yasal istemcilerle bağlantı kuramaz.

Bu saldırıyı yapmak için komut satırına **hping3 -V -c <gönderilmek_istenen_paket_sayısı> -d <paketlerin_boyutu> -S -p <port_numarası> --flood <hedef_ip>** yazarak saldırı başlatılır.

- **-V:** -V parametresi ayrıntılı çıktı almak için kullanılmaktadır.
- **-c <gönderilmek_istenen_paket_sayısı>:** bu komutta saldırı yapılırken kaç tane paket göndereceği belirtilir. Saldırı yapılırken istenilen sayıda paket gönderebilir.
- **-d <paketlerin_boyutu>:** gönderilecek paketlerin boyutunu belirlemek için kullanılır.
- **-S:** -S parametresi, saldırının yapılacağı IP adresine SYN paketlerinin gönderileceğini belirtir.
- **-p <port_numarası>:** saldırı yapılırken paketlerim hangi portun üzerinden göndereceğini belirtir.
- **--flood:** --flood parametresi saldırının yapılacağı bilgisayara paketleri en hızlı biçimde gönderir.

4.2.3. RST/FIN Flood

Bir TCP-SYN oturumunu kapatmak için, istemci ile ana bilgisayar arasında bir RST veya FIN paketi değişimi yapılmalıdır. Bir RST / FIN Flood saldırısı sırasında, sunucunun veri tabanında depolanan oturumların hiçbiriyle bağlantısı olmayan sahte RST veya FIN paketleriyle bombalanır. Kurban sunucunun, gelen paketleri mevcut iletimlerle

karşılaştırmak için çok sayıda sistem kaynağını (ana bellek, işlemci vb.) tüketmesi gerekir; bu da sunucuda üretkenlik kaybı ve kısmen kullanılamamasıyla sonuçlanır.

Bu saldırıyı yapmak için komut satırına **hping3 -FA <hedef_ip> -p <port_numarası> --flood** yazarak saldırı başlatılır.

- **-FA:** -FA parametresi, saldırı yapılacak bilgisayara FIN-ACK paketlerinin gönderilmesini sağlar.
- **-p <port_numarası>:** saldırı yapılırken paketlerin hangi portun üzerinden göndereceğini belirtir.
- **--flood:** --flood parametresi saldırı yapılacak bilgisayara paketleri en hızlı biçimde gönderir.

4.2.4. SYN-ACK Flood

TCP'nin üçlü el sıkışma iletişim sürecinin ikinci adımı bu DDoS saldırısı tarafından sömürülmektedir. Bu adımda, gelen bir SYN paketini kabul etmek için dinleyen ana bilgisayar tarafından bir SYN-ACK paketi oluşturulur. Bir SYN-ACK Flood saldırısında büyük miktarda sahte SYN-ACK paketi hedef sunucuya gönderilir. Saldırı, sunucunun kaynaklarını tüketmeye çalışır.

Bu saldırıyı yapmak için komut satırına **hping3 -SA <hedef_ip> -p <port_numarası> --flood** yazılarak saldırı başlatılmış olur.

- **-SA:** -FA parametresi, saldırı yapılacak bilgisayara SYN-ACK paketlerinin gönderilmesini sağlar.
- **-p <port_numarası>:** saldırıyı yapılırken paketlerin hangi portun üzerinden göndereceğini belirtir.
- **--flood:** --flood parametresi saldırı yapılan hosta paketleri en hızlı biçimde gönderir.

4.2.5. UDP Flood

Adından da anlaşılacağı gibi, bu tip DDoS saldırısında bir sunucu UDP paketleriyle doluyor. TCP'den farklı olarak, istemci ile host arasında iletişim sürecini sona erdirmenin bir sonu yoktur. Bu durum savunma mekanizmalarının UDP Flood saldırısını tanımlamasını zorlaştırmaktadır. Çok sayıda sahte UDP paketi, büyük bir kaynak IP kümesinden hedef sunucuya gönderilir.

UDP Flood saldırıları, rastgele veya belirli bir ağ içindeki sunucunun portu ve IP adreslerini hedef alabilir. Bu tür bir saldırının amacı, tüm mevcut bant genişliği tükenene kadar bir ağdaki bant genişliğini tüketmektir.

Bu saldırıyı yapmak için komut satırına **hping3 <hedef_ip> -I <host_arayüzü> --udp -p <port_numarası> --flood** yazılarak saldırı başlatılmış olur.

- **-I <host_arayüzü>:** saldırıyı yaparken kullanılan arayüzün seçilmesini sağlar.
- **--udp:** saldırıda kullanılacak paketlerin UDP protokolünü kullanmasını sağlar.
- **-p <port_numarası>:** saldırıyı yaparken paketlerin hangi portun üzerinden göndereceğini belirtir.
- **--flood:** --flood parametresi saldırıyı yapılacak hosta paketleri en hızlı biçimde gönderir.

4.2.6. DDoS Saldırısı

DDoS saldırısı, hedeflenen bir sunucunun, hizmetin veya ağın normal trafiğini, Internet trafiğinin akışıyla hedefe ya da etrafındaki altyapısına normal sınırların çok üzerinde yüklenerek bozmayı hedefleyen kötü niyetli bir girişimdir. DDoS saldırıları, birden fazla bilgisayar sistemi saldırı trafiği kaynağı olarak kullanarak gerçekleştirilir. Yüksek düzeyli bir DDoS saldırısı düzenli trafiğin istenen varış yerine ulaşmasını engeller.

Bu saldırıyı yapmak için komut satırına **hping3 --flood <hedef_ip>** yazılarak saldırı başlatılmış olur.

- **--flood:** --flood parametresi saldırının yapılacağı bilgisayara paketleri en hızlı biçimde gönderir.

4.2.7. ARP Poisoning Saldırısı

ARP Poisoning saldırısı man-in-the-middle saldırısının türlerinden bir tanesidir. Türkçe karşılığı "ortadaki adam" olan man-in-the-middle saldırısının amacı, kişilerin hesap bilgilerini ve kredi kartı numaraları gibi kişisel bilgilerini çalmayı amaçlar. Ortadaki adam saldırısı, saldırganın kendisini iki taraf arasındaki sohbete soktuğu, her iki tarafı da taklit ettiği ve iki tarafın birbirine göndermeye çalıştığı bilgilere erişim sağladığı bir tür siber saldırıdır.

Ortadaki adam saldırısına başlamadan önce veya sonrasında host/host veya host/sunucu arasındaki iletişimin devam etmesi için komut satırına **echo 1 > /proc/sys/net/ipv4/ip_forward** yazılır. Eğer bu komut yazılmazsa, kurban cihazlar arasındaki veri alışverişi sonlanır.

ARP Poisoning saldırısı yapılacak hostta iki tane terminal açılır. Bu terminallerden birine
arpspoof -i <host_arayüzü> -t <birinci_hedef_ip> <ikinci_hedef_ip> yazılır, diğer terminale **arpspoof -i <host_arayüzü> -t <ikinci_hedef_ip> <birinci_hedef_ip>** yazılarak saldırı başlatılmış olur.

- **-i <host_arayüzü>:** ARP Spoofing saldırısını yaparken kullanılan arayüzü belirlenmesini sağlar.

- **-t <birinci_hedef_ip> <ikinci_hedef_ip>:** -t <birinci_hedef_ip> saldırının yapılacağı hostun belirlenmesini sağlar. Ardından gelen <ikinci_hedef_ip> ise <birinci_hedef_ip>'ye saldırgan hostu <ikinci_hedef_ip> gibi tanıtmaktadır.

Diğer terminalde hedef IP'lerin yerini değiştirilmesinin sebebi ise ikinci hedeften birinci hedefe giden paketleri kendi üzerimizden aktarılmasını sağlar.

4.3. Saldırıların Yapılışı

Ubuntu sanal makinesine kurulmuş olan Floodlight kontrolcüsü ve Mininet emülatörü YTA oluşturmak için kullanılmıştır. Ubuntu sanal makinesinin IP adresi VirtualBox tarafından 192.168.1.107 olarak atanmıştır. YTA'da oluşturulan topolojide 1 anahtarlayıcı ve 3 tane host bulunmaktadır. Bu hostlar sırasıyla h1, h2 ve h3 olarak Mininet tarafından adlandırılmıştır. Hostların IP adresleri ise sırasıyla 10.0.0.1, 10.0.0.2 ve 10.0.0.3 olarak Mininet tarafından atanmıştır. h1 hostu saldırgan, h2 hostu kurban ve h3 hostu ise h2 hostunun ağdaki durumunu kontrol etmek amacıyla ping atacak olan host olarak kullanılmıştır.

Geleneksel bilgisayar ağlarında yapılacak saldırı için Kali Linux sanal makinesinin IP adresi VirtualBox tarafından 192.168.1.106 olarak atanmıştır. Kurban olarak belirlenen ana bilgisayarın IP adresi ise 192.168.1.105'tir.

Hem YTA hem de geleneksel bilgisayar ağlarında yapılan DDoS saldırında kullanılacak parameterler olan **<rastgele_girilen_ip>, <port_numarası>, <gönderilmek_istenen_paket_sayısı> ve <paketlerin_boyutu>** değerleri rastgele olarak atanmıştır. **<host_arayüzü>** ise saldırının yapılacağı hostun arayüzüne göre değişkenlik göstermiştir. Saldırı yapılmaya başladıktan sonra kurban hostun ağdaki durumunu kontrol etmek için kontrolcü hosttan kurban hosta ping atılmıştır.

Geleneksel bilgisayar ağlarında ARP Posioning için hem Ubuntu sanal makinesi hem de ana bilgisayar kullanılmıştır. ARP Poisoning saldırında kurban hostların birbirlerini ARP tablolarına eklemesi için kendi aralarında ping atılmıştır. Daha sonrasında saldırıyı yapacak host olan Kali Linux terminale geçip aralarındaki iletişimin kopmaması için ilk başta **echo 1 > /proc/sys/net/ipv4/ip_forward** yazılarak saldırı başladıktan sonra iletişim devam etmesi sağlanmıştır. En sonunda ise saldırgan hostun terminallerden birine **arpspoof -i <host_arayüzü> -t <birinci_hedef_ip> <ikinci_hedef_ip>,** diğerine ise **arpspoof -i <host_arayüzü> -t <ikinci_hedef_ip> <birinci_hedef_ip>** yazılarak kurban hostlar arasında iletişimin saldırgan host üzerinden geçmesi sağlanmıştır.

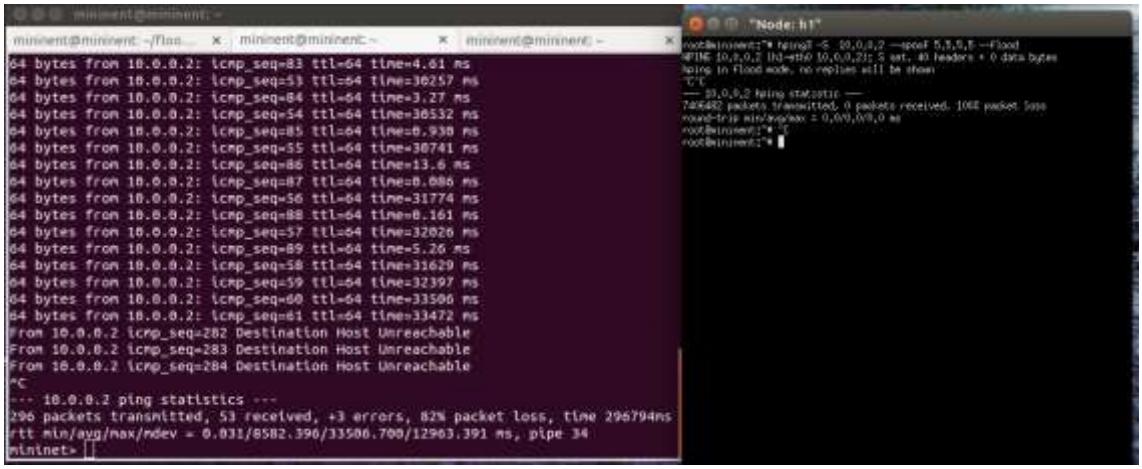
BÖLÜM 5

SONUÇLAR VE TARTIŞMA

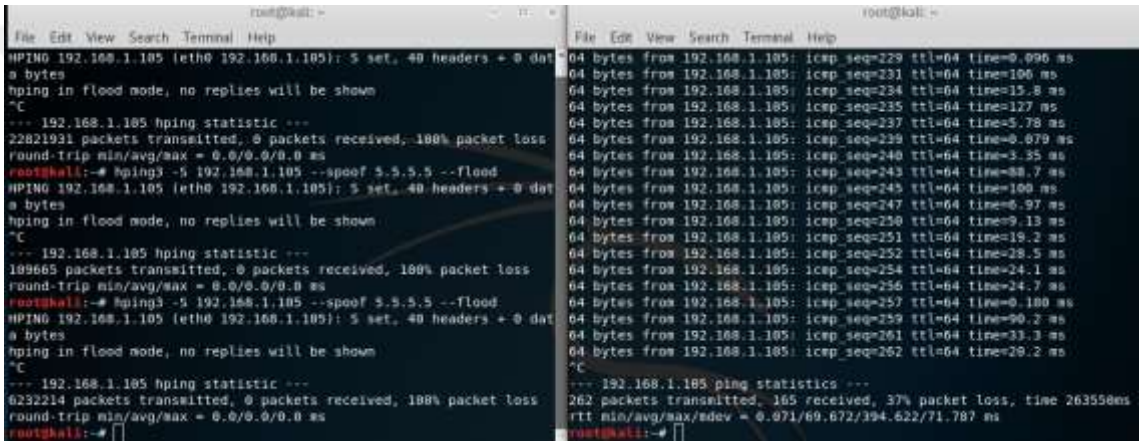
Bölüm 4'te anlatılan siber saldırılar hem geleneksel bilgisayar ağlarına hem de yazılım tanımlı ağlarına uygulanmıştır. Bu saldırıların sonuçları konsol ekranında elde edilmiş olup bu bölümde anlatılmıştır.

5.1. IP Spoofing Saldırısı Sonuçları

YTA'da saldırıya başladıktan kısa bir süre sonra h3'ten gönderilen ping paketleri hedefine ulaşmamaya başlamıştır. Daha sonra tekrardan paketleri almaya devam etmiş fakat en sonunda h2 hostu ağdan düşmüştür (Yavuz ve Tuna, 2019). Şekil 5.1 a'da saldırının sonucu gösterilmektedir. Geleneksel bilgisayar ağlarında yapılan saldırılar YTA'nın aksine başarılı olmamıştır. Yapılan saldırı sadece ana bilgisayara gönderilen paketlerin bazılarının düşmesine neden olmuştur. Şekil 5.2 b'de gerçekleştirilen saldırının sonucu gösterilmektedir.



a) Yazılım Tanımlı Ağ



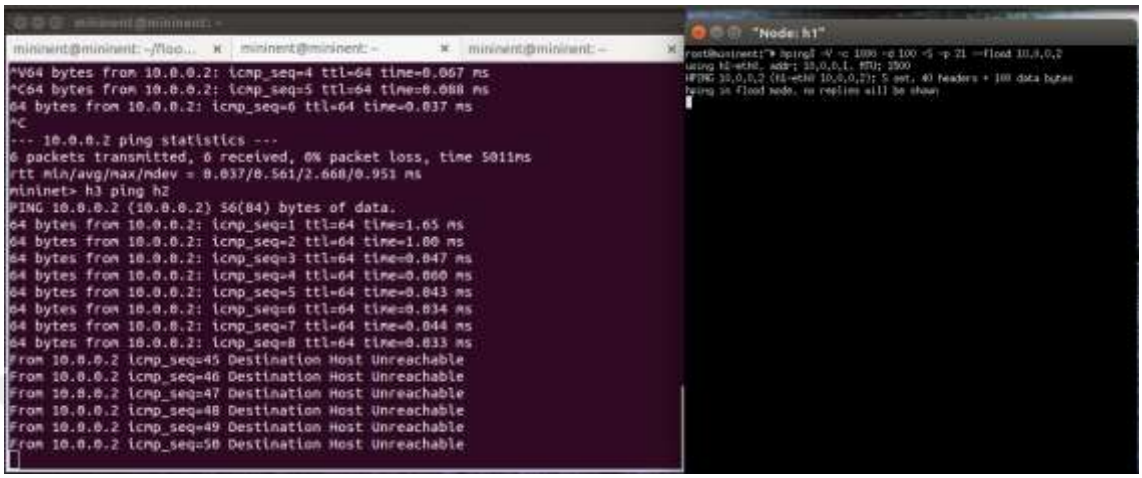
b) Geleneksel Bilgisayar Ağı

Şekil 5.1. IP Spoofing Saldırı Sonuçları

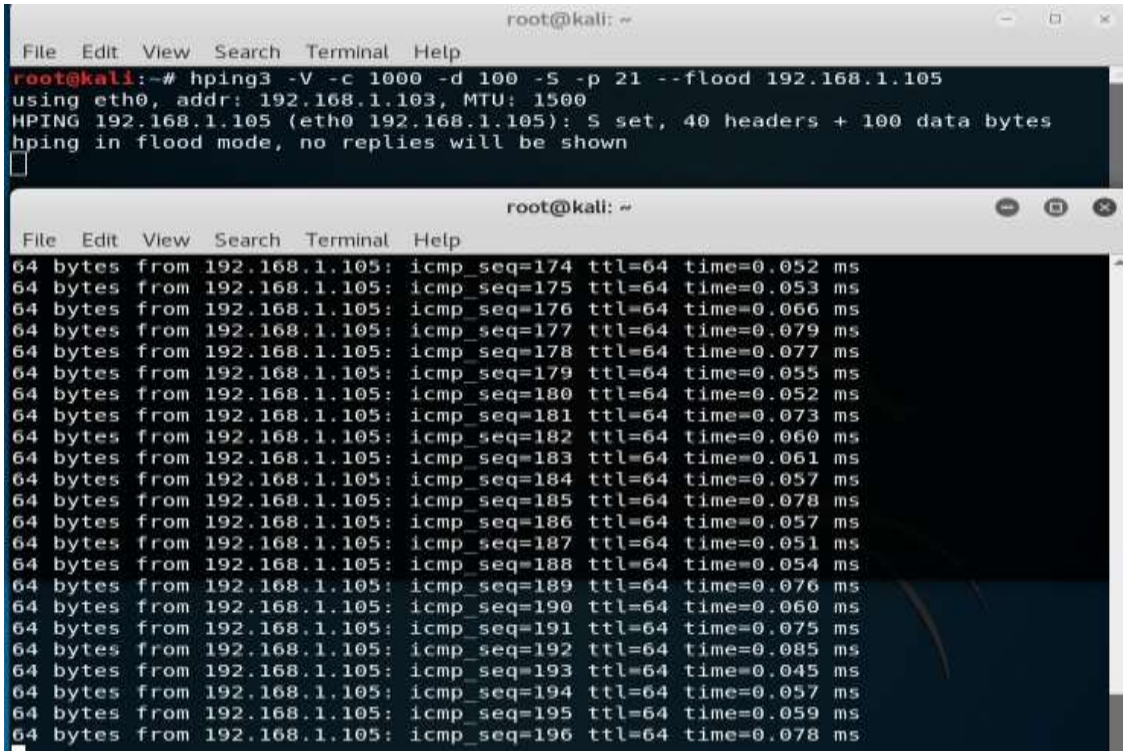
5.2. SYN Flooding Saldırısı Sonuçları

SYN Flooding saldırısı yazılım tanımlı ağlarda h2 hostunun ağdan düşmesi ile sonuçlanmıştır (Yavuz ve Tuna, 2019). h2 hostu bazen ağdan düştükten belirli bir sonra ağa tekrardan bağlanmıştır. Devam eden saldırı sonucunda h2 hostu ağdan tamamen düşmüştür. Bazen de h2 hostu ağdan düştükten sonra ağa hiç bağlanamamıştır. Şekil 5.2 a'da saldırının sonucu gösterilmektedir.

Geleneksel bilgisayar ağları yapılan SYN Flooding saldırısından etkilenmemiştir. Saldırı esnasında kontrol amacıyla gönderilen paketlerde de herhangi bir kayıp olmamıştır. Saldırının sonucu Şekil 5.2 b'de gösterilmektedir.



a) Yazılım Tanımlı Ağ



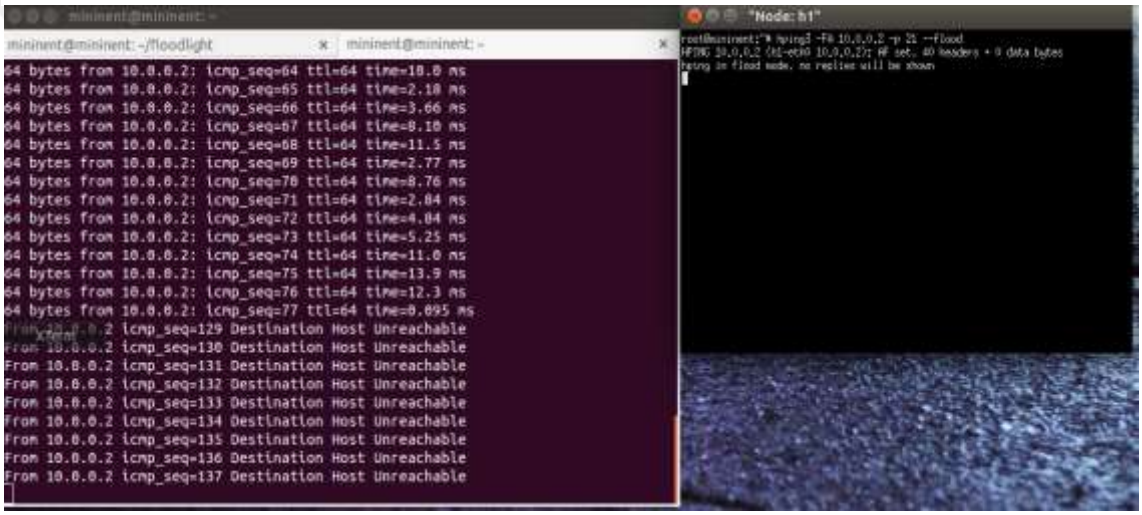
b) Geleneksel Bilgisayar Ağı

Şekil 5.2. SYN Flooding Saldırısı Sonuçları

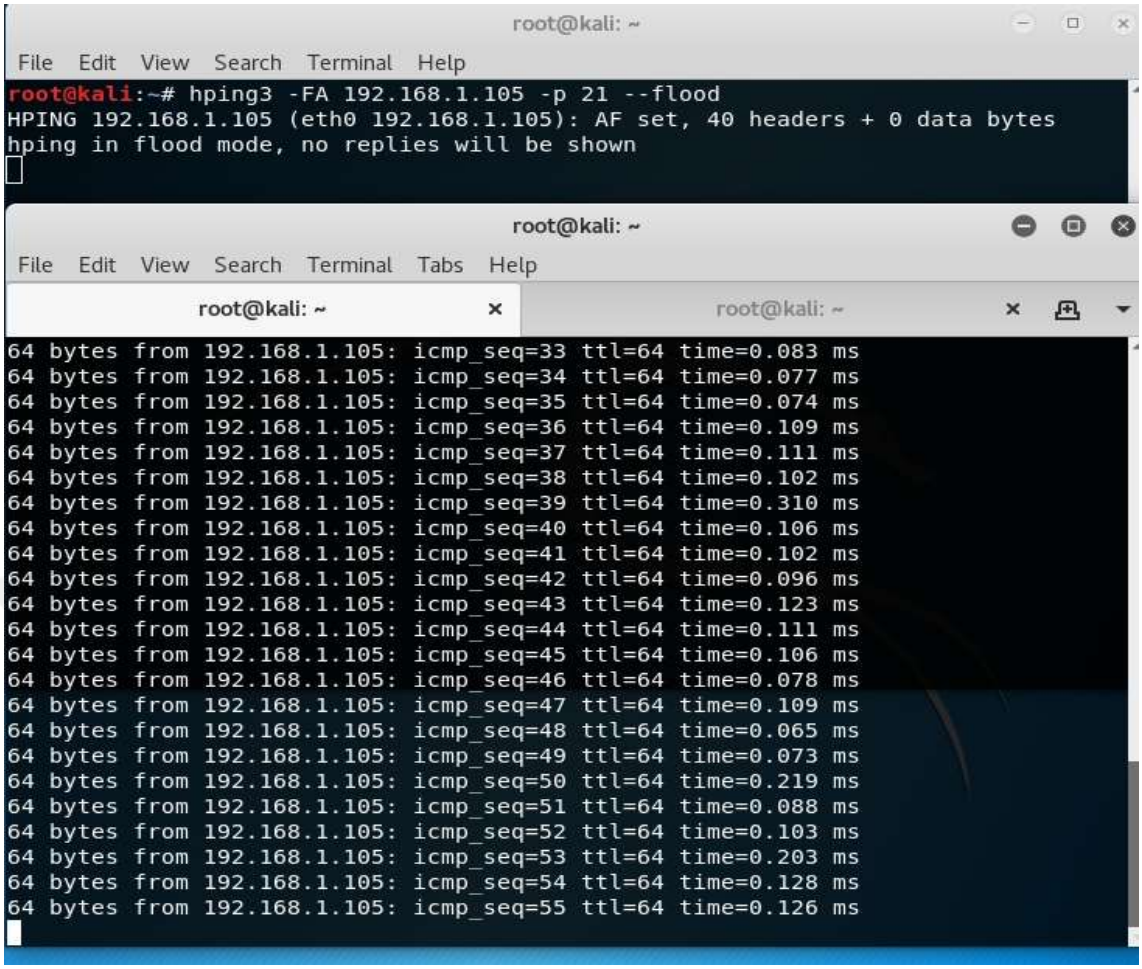
5.3. RST/FIN Flood Saldırısı Sonuçları

YTA'da yapılan RST/FIN Flood saldırısı h2 hostunun ağdan düşmesi ile sonuçlanmıştır. H2 hostu bazen ağdan düştükten sonra birkaç kez ağa bağlanıp tekrardan düşmüştür. Fakat en sonunda ağdan tamamen düşmüştür. Bazen de h2 hostu ağdan düştükten sonra bir daha ağa bağlanamamıştır (Yavuz ve Tuna, 2019). Saldırının sonucu Şekil 5.3 a'da gösterilmektedir.

Geleneksel bilgisayar ağları yapılan RST/FIN Flood saldırılarından etkilenmemiştir. Saldırı esnasında kontrol amaçlı gönderilen ping paketlerinde de herhangi bir kayıp olmamıştır. Yapılan saldırının sonucu Şekil 5.3 b'de gösterilmektedir.



a) Yazılım Tanımlı Ağ



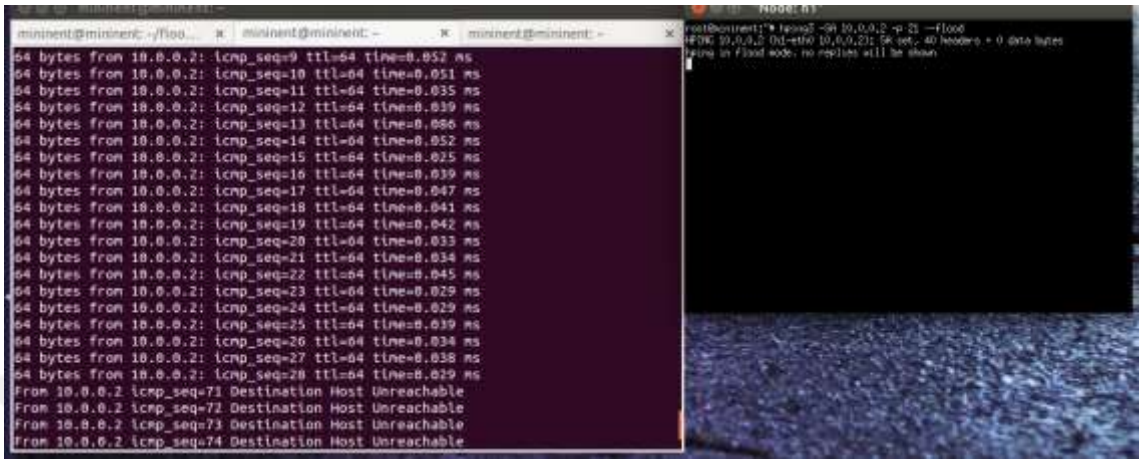
b) Geleneksel Bilgisayar Ağı

Şekil 5.3. RST/FIN Flood Saldırısı Sonuçları

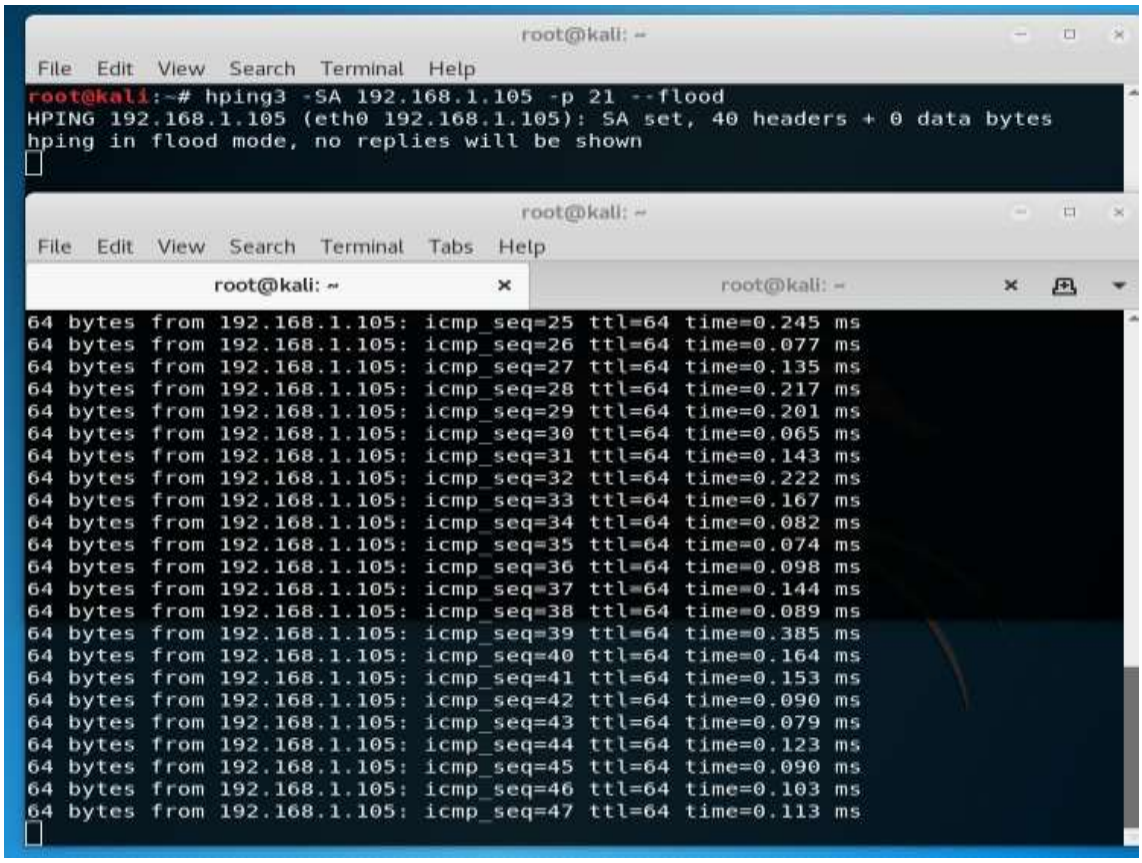
5.4. SYN-ACK Flood Saldırısı Sonuçları

YTA'da yapılan bu saldırıda da 2 şekilde sonuçlanmıştır. İlk durumda, saldırı başladıktan bir süre sonra h2 hostu ağdan hemen kopmuştur. Diğer durumda ise, saldırı başladıktan sonra h2 host bir süreliğine ağdan kopmuş daha sonrasında ise ağa tekrardan bağlanmıştır (Yavuz ve Tuna, 2019). Fakat en sonunda ise h2 hostu ağdan tamamen kopmuştur. Şekil 5.4 a'da saldırının sonucu gösterilmektedir.

Geleneksel bilgisayar ağlarında SYN-ACK Flood saldırı başladıktan sonra ana bilgisayara kontrol amaçlı gönderilen paketlerde herhangi bir kayıp olmamış ve ana bilgisayar saldırıdan etkilenmemiştir. Saldırının sonucu Şekil 5.4 b'de gösterilmektedir.



a) Yazılım Tanımlı Ağ



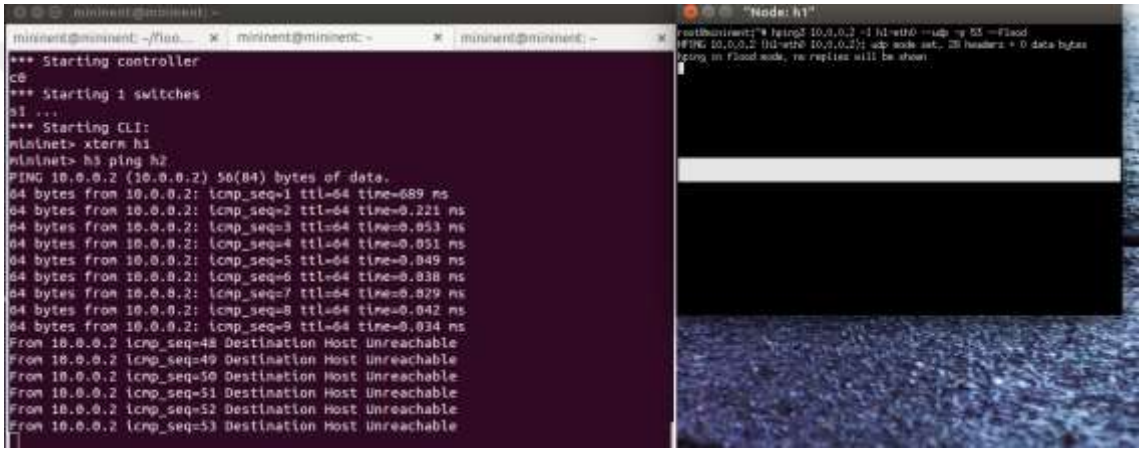
b) Geleneksel Bilgisayar Ağı

Şekil 5.4. SYN-ACK Flood Saldırısı Sonuçları

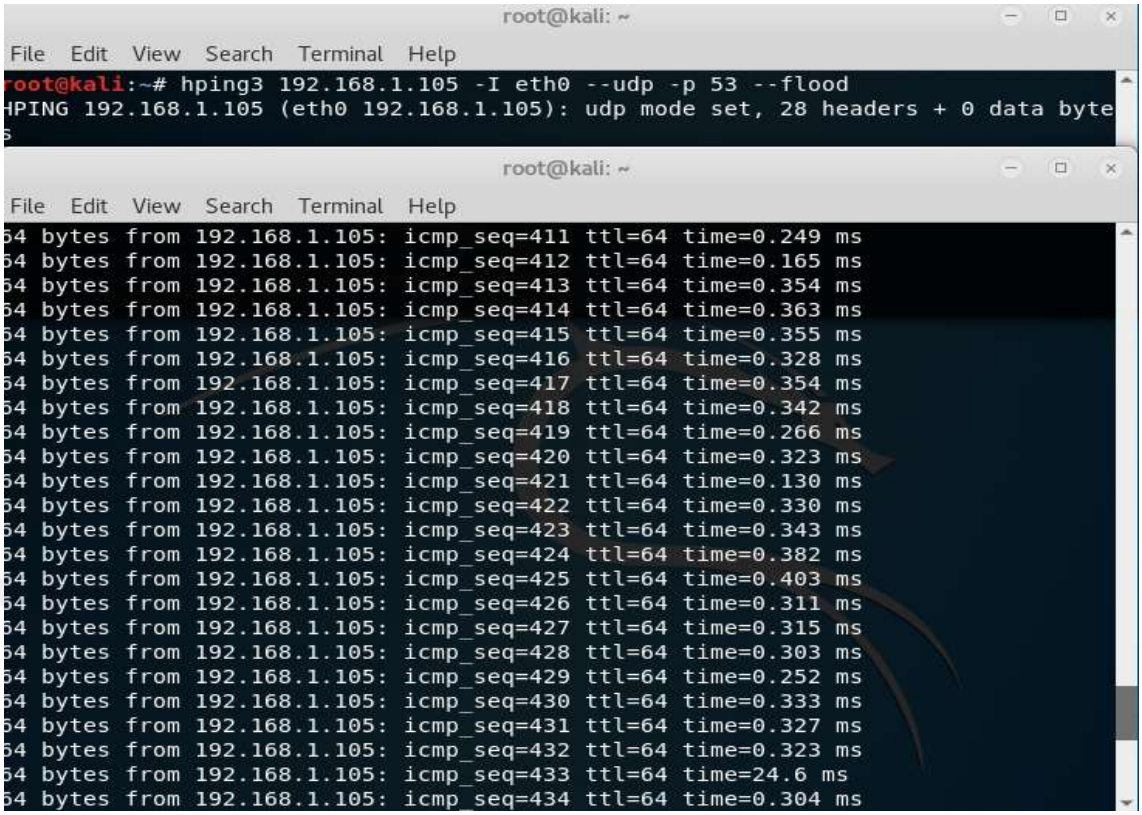
5.5. UDP Flood Saldırısı Sonuçları

YTA'da UDP Flood saldırısı başladıktan kısa bir süre sonra h2 hostunun ağdan düşmesi ile sonuçlanmıştır. h2 hostu bazen ağdan düştükten sonra ağa tekrar bağlanmış fakat en sonunda ağdan tamamen düşmüştür (Yavuz ve Tuna, 2019). Şekil 5.5 a'da saldırının sonucu gösterilmektedir.

Geleneksel bilgisayar ağlar yapılan UDP Flood saldırılarından etkilenmemiştir. Saldırı esnasında kontrol amaçlı gönderilen ping paketlerinde de herhangi bir kayıp olmamıştır. Saldırının sonucu Şekil 5.5 b'de gösterilmektedir.



a) Yazılım Tanımlı Ağ



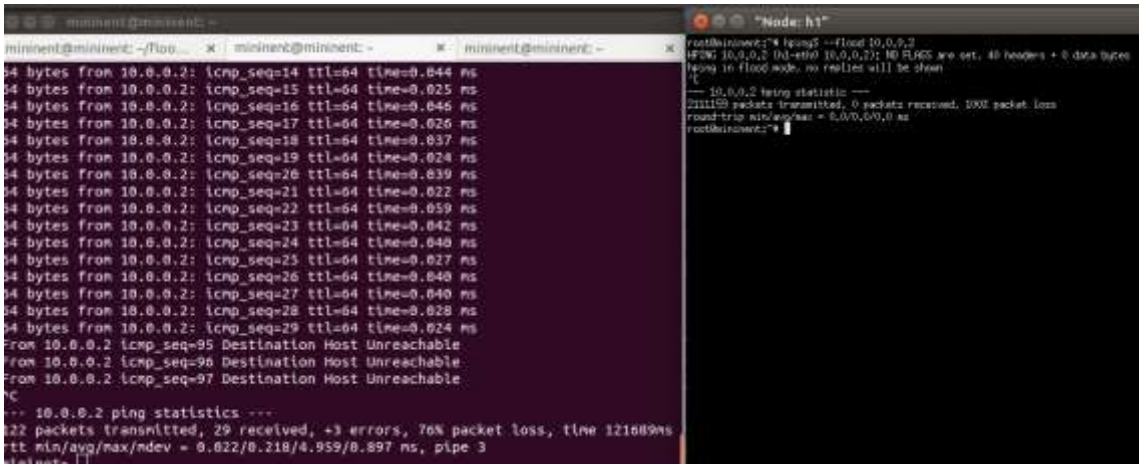
b) Geleneksel Bilgisayar Ağı

Şekil 5.5. UDP Flood Saldırısının Sonuçları

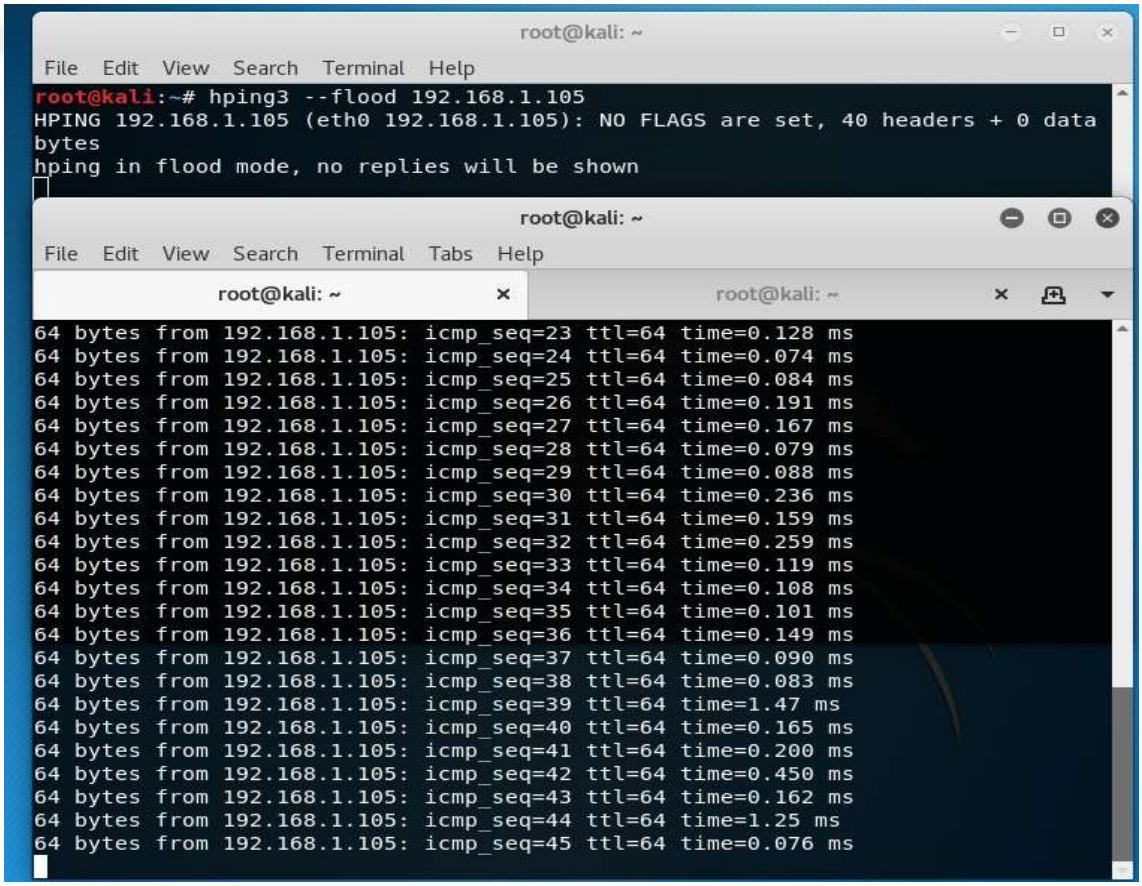
5.6. DDOS Saldırısının Sonuçları

YTA'da DDoS saldırı başladıktan kısa bir süre sonra h2 hostu ağdan düşmüştür ve ağa bir daha bağlanamamıştır (Yavuz ve Tuna, 2019). Şekil 5.6 a'da saldırının sonucu gösterilmektedir.

Geleneksel bilgisayar ağlarında yapılan DDoS saldırısında ana bilgisayar bu saldırıdan herhangi bir şekilde etkilenmemiş ve kontrol amacıyla gönderilen paketlerde kayıp olmamıştır. Şekil 5.6 b'de saldırının sonucu gösterilmektedir.



a) Yazılım Tanımlı Ağlar

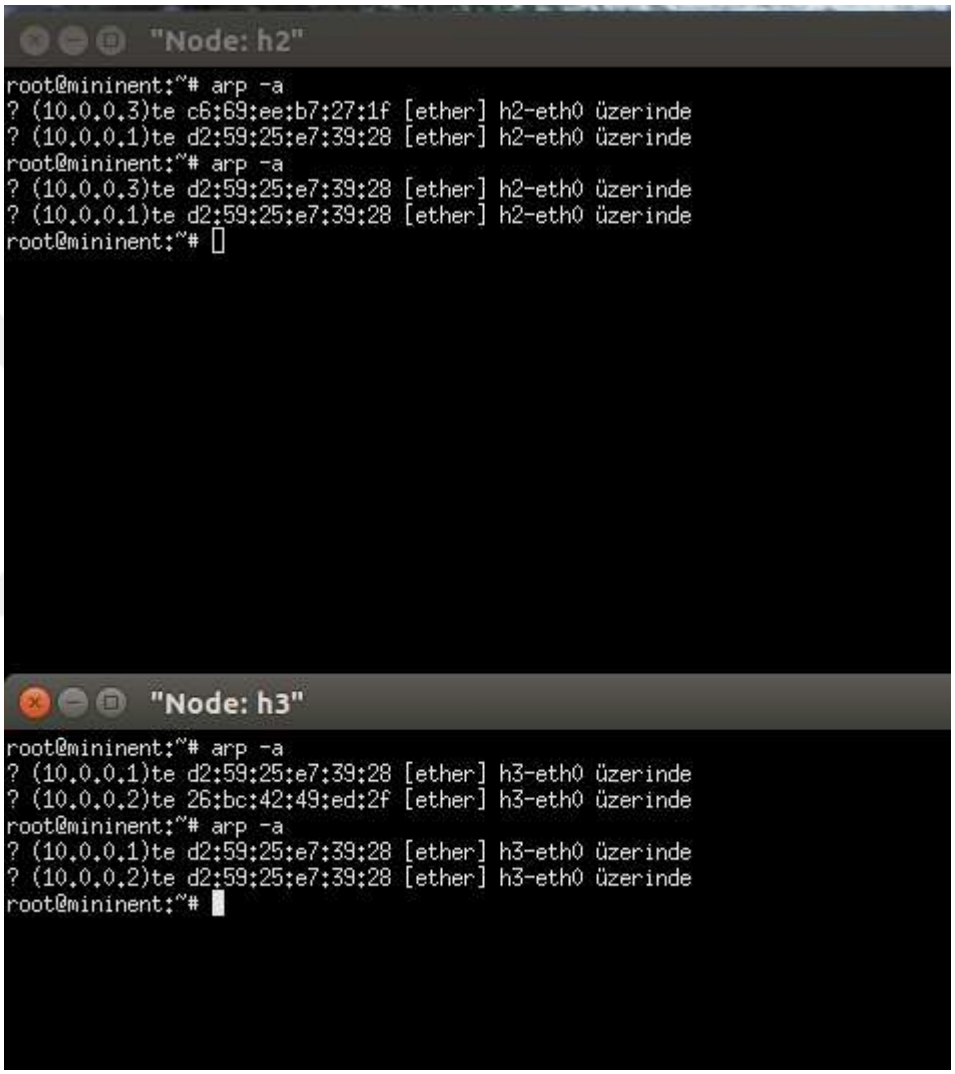


b) Geleneksel Bilgisayar Ağı

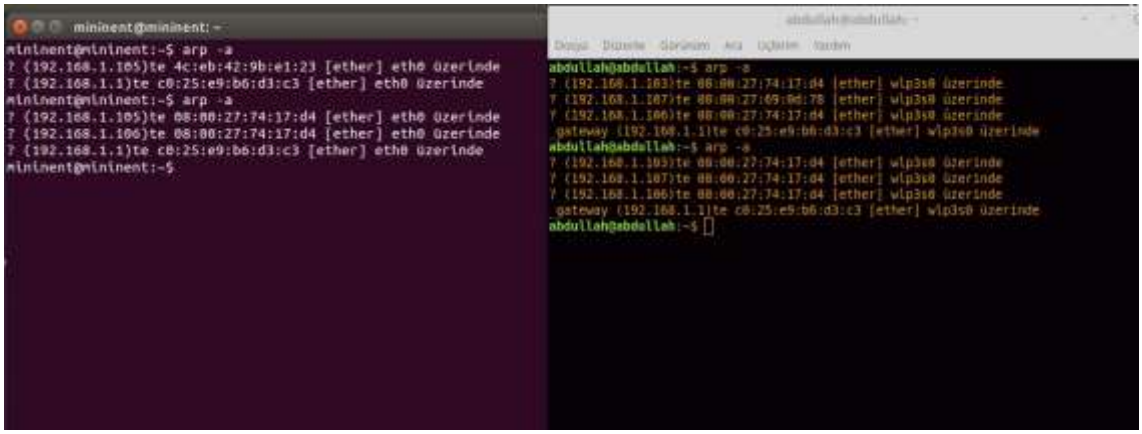
Şekil 5.6. DDoS Saldırısı Sonuçları

5.7. ARP Poisoning Saldırısı Sonuçları

ARP Poisoning saldırısı hem yazılım tanımlı ağlarda hem de geleneksel bilgisayar ağlarında başarılı olmuştur (Yavuz ve Tuna, 2019). YTA'da ARP Poisoning saldırısının sonucu Şekil 5.7 a'da, geleneksel bilgisayar ağlarında ARP Poisoning saldırının sonucu Şekil 5.7 b'de gösterilmiştir.



a) Yazılım Tanımlı Ağ



b) Geleneksel Bilgisayar Ağı

Şekil 5.7. ARP Poisoning Saldırısının Sonuçları

5.8. Değerlendirme

Yazılım tanımlı ağlar birçok avantaja sahip olmakla birlikte bilgi güvenliği tehditleri açısından yeterince ele alınmamıştır. Bu nedenle yazılım tanımlı ağların bilgi güvenliği tehditleri açısından değerlendirilmesi büyük önem taşımaktadır. Bu çalışmada yazılım tanımlı ağların bilgi güvenliği tehditlerine karşı zafiyetleri, geleneksel bilgisayar ağlarıyla kıyaslanarak değerlendirilmiş ve çözüm önerileri sunulmuştur. Hping3 ve Dsniff uygulamaları kullanılarak gerçekleştirilen benzetim çalışmalarında IP Spoofing, SYN Flood, RST/FIN Flood, SYN-ACK Flood, UDP Flood, ARP Poisoning ve DDoS saldırıları gerçekleştirilmiştir.

Gerçekleştirilen benzetim çalışmalarının sonuçları, geleneksel bilgisayar ağlarına kıyasla yazılım tanımlı ağların daha fazla güvenlik zafiyeti taşıdığı göstermektedir. Öte yandan, yapılan saldırıları engellemek amacıyla ağ için veri hızı sınırlaması, paket gönderim sınırlaması ve kimlik doğrulama teknikleriyle güvenlik duvarları ve saldırı tespit ve önleme sistemlerinin kurulumu gibi tedbirler alınarak yazılım tanımlı ağlarda bulunan güvenlik zafiyetleri ortadan kaldırılabilir.

KAYNAKLAR

Ahmad, I., Namal, S., Ylianttila, M., & Gurtov, A. (2015). Security in software defined networks: A survey. *IEEE Communications Surveys & Tutorials*, *17*(4), 2317-2346.

Alparslan, Ö. (2016). Veri Merkezlerinde Büyük Boyutlu Verilerin Taşınması Sürecinde Yazılım Tanımlı Ağların (SDN) Kullanımı. Yüksek Lisans Tezi. Gazi Üniversitesi

Bavier, A., Feamster, N., Huang, M., Peterson, L., & Rexford, J. (2006, August). In VINI veritas: realistic and controlled network experimentation. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications* (pp. 3-14).

Caesar, M., Caldwell, D., Feamster, N., Rexford, J., Shaikh, A., & van der Merwe, J. (2005, May). Design and implementation of a routing control platform. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2* (pp. 15-28).

Casado, M., Garfinkel, T., Akella, A., Freedman, M. J., Boneh, D., McKeown, N., & Shenker, S. (2006, August). SANE: A Protection Architecture for Enterprise Networks. In *USENIX Security Symposium* (Vol. 49, p. 50).

Casado, M., Freedman, M. J., Pettit, J., Luo, J., McKeown, N., & Shenker, S. (2007). Ethane: Taking control of the enterprise. *ACM SIGCOMM computer communication review*, *37*(4), 1-12.

Cemal Taner. (2018). https://www.cemaltaner.com.tr/2018/09/20/tcp-ip-protokol-kumesi-nedir/ Erişim Tarihi: 02.10.2019

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., & Bowman, M. (2003). Planetlab: an overlay testbed for broad-coverage ervices. *ACM SIGCOMM Computer Communication Review*, *33*(3), 3-12.

De Oliveira, R. L. S., Schweitzer, C. M., Shinoda, A. A., & Prete, L. R. (2014, June). Using mininet for emulation and prototyping software-defined networks. In *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)* (pp. 1-6). IEEE.

Dhaya R., Maharaj S., Sownmya J., Kanthavel R. (2017). Software Defined Networking: Viewpoint of From IP Networking, PROS and CONS and Exploration Thoughts. *International Conference on Intelligent Computing and Control Systems (ICICCS)*. https://doi.org/10.1109/ICCONS.2017.8250627

Floodlight. (2012). https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview Erişim Tarihi: 30.10.2019

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., & Shenker, S. (2008). NOX: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, *38*(3), 105-110.

Hping. (2006). http://www.hping.org Erişim Tarihi: 06.10.2019

ISO, I. (1994). IEC 7498-1: 1994 information technology–open systems interconnection–basic reference model: The basic model. *ISO standard ISO/IEC*, 7498-1.

Kali. (2013). https://www.kali.org Erişim Tarihi: 30.10.2019

Kaur, K., Singh, J., Ghumman N. (2014). Mininet as Software Defined Networking Testing Platform. *International Conference on Communication, Computing & Systems*, 139-142.

King, D., Rotsos, C., Aguado, A., Georgalas, N., Lopez, V. (2016). The Software Defined Transport Network: Fundamentals, findings and futures, *2016 18th International Conference on Transparent Optical Networks (ICTON)*. https://doi.org/10.1109/ICTON.2016.7550669

Kizza, J. M.. (2009). *Guide to computer network security*. London: Springer.

Kreutz D., Ramos F., Verissimo P., Rothenberg C. E., Azodolmolky S., and Uhlig S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE,* 1, 14-76.

Lazar, K., Lim, S., and Marconcini, F. (1996). Realizing a foundation for programmability of ATM networks with the binding architecture. *Journal on Selected Areas in Communications*, 14, 1214–1227.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38, 69-74.

Macedonia, M. R., & Brutzman, D. P. (1994). MBone provides audio and video across the Internet. *Computer*, *27*(4), 30-36.

Mininet. (2018). mininet.org Erişim Tarihi: 30.09.2019

Monkey. (2020). https://www.monkey.org/~dugsong/dsniff/ Erişim Tarihi: 06.10.2019

Opennetworking. (2013). https://www.opennetworking.org Erişim Tarihi: 03.10.2019

Opennetworking. (2014). https://www.opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf Erişim Tarihi: 27.09.1019

Peterson, L., Anderson, T., Blumenthal, D., Casey, D., Clark, D., Estrin, D., ... & Shenker, S. (2006). GENI design principles.

Sahba, R. (2018). A Brief Study of Software Defined Networking for Cloud Computing. *2018 World Automation Congress, 6-9.*

Salman, O., Elhajj, I. H., Kayssi, A., & Chehab, A. (2016, April). SDN controllers: A comparative study. In *2016 18th Mediterranean Electrotechnical Conference (MELECON)* (pp. 1-6). IEEE.

Sdxcentral. (2014). https://www.sdxcentral.com/networking/sdn/definitions/sdn-controllers/ Erişim Tarihi: 04.10.2019

Searchnetworking. (2018). https://searchnetworking.techtarget.com/definition/SDN-controller-software-defined-networking-controller Erişim Tarihi 04.10.2019

Searchnetworking. (2013). https://searchnetworking.techtarget.com/definition/OpenFlow-controller Erişim Tarihi: 25.09.2019

Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., Viljoen N., Miller N. M., Rao, N. (2013). Are We Ready for SDN? Implementation Challenges for Software-Defined Networks. *IEEE Communications Magazine*, 51, 36-43

Sheinbein, D., Weber, R. P. (1982). 800 Service Using SPC Network Capability. *Bell System Techonology Journal*, 61, 1737–1744

Socolofsky, T., & Kale, C. (1991). Rfc 1180-tcp. IP tutorial, 10.Tennenhouse, D.,

Smith, J., Sincoskie W., Wetherall D., and Minden, G. (1997). A survey of active network research. *IEEE Communication Magazine*, 35, 80–86

Van der Merwe, J. E., Rooney, S., Leslie, L., & Crosby, S. (1998). The Tempest-a practical framework for network programmability. *IEEE network*, *12*(3), 20-28.

Yavuz, A., & Tuna, G. BİLGİ GÜVENLİĞİ TEHDİTLERİ: YAZILIM TANIMLI AĞLAR GELENEKSEL BİLGİSAYAR AĞLARINA KARŞI. *Kırklareli Üniversitesi Mühendislik ve Fen Bilimleri Dergisi*, *5*(2), 200-209.

Xia, W., Wen, Y., Foh, C. H., Niyato, D., & Xie, H. (2014). A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, *17*(1), 27-51.

ÖZGEÇMİŞ

Abdullah YAVUZ 1994 yılında Edirne'nin Meriç ilçesinde doğdu. İlköğretimini Kırklareli'nin Kofçaz ilçesinde, ortaöğrenimi Kırklareli'nde tamamladı. 2012 yılında Kırklareli Anadolu Lisesi'nden mezun oldu. Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nden 2017 yılında mezun oldu. 2018 yılında Trakya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde yüksek lisans eğitimine başladı. 2020 yılında İstanbul Rumeli Üniversitesi Bilgisayar Mühendisliği Bölümü'nde araştırma görevlisi olarak çalışmaya başladı ve görevine hala devam etmektedir.

TEZ ÖĞRENCİSİNE AİT TEZ İLE İLGİLİ BİLİMSEL FAALİYETLER

1) Ulusal Hakemli Dergilerde Yayınlanan Makaleler

- Yavuz, A., & Tuna, G. BİLGİ GÜVENLİĞİ TEHDİTLERİ: YAZILIM TANIMLI AĞLAR GELENEKSEL BİLGİSAYAR AĞLARINA KARŞI. *Kırklareli Üniversitesi Mühendislik ve Fen Bilimleri Dergisi*, *5*(2), 200-209.