

**T.C.**  
**FIRAT ÜNİVERSİTESİ**  
**TEKNOLOJİ FAKÜLTESİ**  
**ADLİ BİLİŞİM MÜHENDİSLİĞİ BÖLÜMÜ**

# **YEREL AĞDA SALDIRI VE AĞ TARAMA TESPİTİ**

**BİTİRME PROJESİ**

**HAZIRLAYAN**  
**YUNUS EMRE DEMİRBAŞ**

**Bitirme Yöneticisi**  
**Dr. Öğr. Üyesi Erhan AKBAL**

**ELAZIĞ 2019**



**T.C.**  
**FIRAT ÜNİVERSİTESİ**  
**TEKNOLOJİ FAKÜLTESİ**  
**ADLİ BİLİŞİM MÜHENDİSLİĞİ BÖLÜMÜ**

**YEREL AĞDA SALDIRI VE AĞ TARAMA TESPİTİ**

**BİTİRME PROJESİ**

**HAZIRLAYAN**

**Yunus Emre DEMİRBAŞ**

Bu proje, ..... tarihinde aşağıda belirtilen jüri tarafından oybirliği /oyçokluğu ile başarılı / başarısız olarak değerlendirilmiştir.

Danışman: Dr. Öğr. Üyesi Erhan AKBAL

## ÖNSÖZ

Yapmış olduğum bu proje, meslek hayatımda içinde bulunmak istediğim Ar-Ge çalışmaları için bir ön hazırlık olmuştur. Çalışmanın dar bir kapsamı bulunmaktadır. Çalışmalarım sonucunda öğrenmiş olduğum bilgi ve yetkinlikler sayesinde çalışma hayatına kolay bir şekilde uyum sağlayabileceğimi görmekteyim. Ülkeme faydalı bir vatandaş olma, katma değeri yüksek olan projelerde yer alma ümidiyle. Aynı zamanda;

Üniversite hayatım boyunca derslerine büyük bir zevkle gittiğim, duruşuyla ve davranışlarıyla her zaman öğrencilerinin takdirini kazanmış, derslerinde tecrübelerini de aktaran ve vizyonumun gelişmesine katkı sunan çok kıymetli proje danışman hocam Dr. Öğr. Üyesi Erhan AKBAL’a sonsuz teşekkürlerimi sunarım. Aynı zamanda proje fikrinin oluşmasında katkı sağlayan, projeyi geliştirme sürecinde bilgisinden ve tecrübelerinden faydalandığım, güler yüzünü eksik etmeyen, her zaman büyük bir heyecan ve zevkle bir şeyler öğretmeye çalışan iş yeri eğitim sorumlum PTT Bilgi Güvenliği Daire Başkanı Ömer ELMASOĞLU’na da sonsuz teşekkürlerimi sunarım.

Son olarak, bölümümüzün çok kıymetli akademisyenlerine, bana vermiş oldukları değerli bilgiler için ve her zaman bir abi/abla sıcaklığıyla davrandıkları, bölümümü bir aile gibi hissetmemi sağladıkları için sonsuz teşekkürlerimi sunuyorum. Beni bu günlere getiren ve desteklerini hiçbir zaman esirgemeyen çok kıymetli aileme de teşekkürü bir borç bilirim.

Yunus Emre DEMİRBAŞ

# İçindekiler

Özet .....	iv
1.Giriş .....	1
2.Materyal ve Metot .....	1
2.1 Kullanılan Yazılım Dili .....	1
2.2 Kullanılan kütüphaneler .....	1
3. Proje Yapım Aşaması ve Yöntemi .....	2
3.1. Network Sniffing, Cihaz Kurulumu ve Kullanımı İçin Hazırlıklar .....	2
3.2. Thread Araştırması, Regex İle İp ve MAC Eşleme .....	4
3.3. Python Socket Programlama .....	5
3.4. Veri Tabanı Tasarımı .....	6
3.5. Kullanılacak Veri Tabanı ve Tabloların Oluşturulması .....	8
3.6. Global Değişkenler, Python Kütüphane Oluşturma, SQLite Database .....	9
3.7. Wireshark il UDP Paket İçeriğini Analiz Etme, Nmap UDP Tarama .....	11
3.8. UDP Port Tarama Tespit Kodunun Yazılması, Mitm Araştırma .....	13
3.9. Python SMTP Mail Gönderme .....	18
3.10 Log Dosyası Oluşturma .....	21
4. Sonuçlar .....	22
5. Öneriler .....	22
6. Kaynakça .....	23
7. Ekler .....	25

## ŞEKİLLER TABLOSU

Şekil 1 ip regex yapısı.....	4
Şekil 2 MAC regex yapısı.....	5
Şekil 3 Socket program kodu.....	5
Şekil 4 Socket program kodunun ham çıktısı .....	6
Şekil 5 mysql kullanıcı oluşturma .....	7
Şekil 6 mysql yetkiler tablosu.....	7
Şekil 7 mysql kullanıcı yetki çıktısı.....	8
Şekil 8 mysql tablo oluşturma sorgusu .....	8
Şekil 9 veri tabanı tablosu genel görünüm.....	9
Şekil 10 global değişkenlerin test edilmesi.....	10
Şekil 11 google dns UDP taraması .....	12
Şekil 12 UDP tarama sonucunun incelenmesi .....	13
Şekil 13 UDP taramayı tespit eden kod .....	13
Şekil 14 websploit help çıktısı .....	15
Şekil 15 websploit modüller çıktısı .....	16
Şekil 16 websploit options çıktısı .....	17
Şekil 17 mitm saldırı sonrası veri tabanı kayıtları .....	17
Şekil 18 mitim saldırı tespit kodu .....	18
Şekil 19 sistem yöneticisine mail gönderen python fonksiyonu.....	19
Şekil 20 alert mesajını oluşturan message değişkeni.....	20
Şekil 21 mailin gönderildiğine dair ekran görüntüsü.....	20
Şekil 22 mailin gelen kutusundaki ekran görüntüsü .....	21
Şekil 23 log dosyasını oluşturan python fonksiyonu .....	21

## ÖZET

Yapılan çalışma, yerel ağ üzerindeki trafiğin dinlenerek, trafik üzerinde herhangi bir saldırı amaçlı ya da saldırı için zemin hazırlayacak bir davranışın olup olmadığı tespit edilmeye çalışılmıştır. Proje tamamında yerel ağ üzerinde UDP port tarama ve Ortadaki Adam Saldırısı(MITM) tespiti yapılmaktadır. Projede python programlama dili kullanılmıştır. Debian tabanlı işletim sistemlerinde kullanılabilen scapy toolu, python kütüphanesi olarak kullanılmış, ağ trafiğini scapy ile ayrıştırmış bulunmaktayım. Proje iki ana process üzerindedir. Bunlardan birincisi, “sniffing.py” isimli python dosyasıdır. Bu dosyada paketler, ethernet tiplerine ve protokollerine ayrıştırılmakta ve veri tabanına kaydedilmektedir. İkinci ana process olan “saldiritespit.py” ise; veri tabanına kaydedilen verileri, tespit edilmek istenen saldırı ya da tarama tiplerine göre gruplandırıp belirlenen kriterlere göre kıyas yapmaktadır. Program içerisindeki belirlenen durumlara göre de aksiyon almaktadır. Daha öncede belirtildiği gibi program şu anlık UDP taramalarını ve MITM saldırılarını tespit edebilmekte. Tespit işleminden sonra kritiklik seviyesi belirlenmekte. Kritiklik seviyesine göre de veri tabanına kaydederek sistem üzerindeki güvenlik duvarına saldırgan ip adresini kaydedip, olası saldırı ihtimalinden korunulmaktadır. Herhangi bir güvenlik derecesine girmiş olan istekler ise “saldiritespit.py” dosyasının bulunduğu dizine “log\_file.log” dosyası oluşturup içerisine bağlantı istekleri kaydedilmektedir. Ayrıca kritik olarak nitelendirilmiş bağlantı tespit edildiğinde, sistem içerisinde belirtilen e-posta adreslerine ip bilgileri ve saldırı tipleri bilgi maili olarak gönderilmektedir. Geliştirme süreci performans açısından, VMware sanal makinelerinde yapılmış olup daha sonra raspberry cihaz üzerinde çalışmaya hazır hale getirilmiştir.

Anahtar Kelimeler: Yerel ağ trafik analizi, ağ tarama, saldırı tespit, scapy ile paket analizi

## 1. GİRİŞ

Bilgisayarlar, veri alışverişini gönderdikleri paketler aracılığıyla yapmaktadır. Bu paketlerin içerikleri, etiketleri, boyutları, başlıkları, vs. birbirinden farklılık gösterebilmektedir. Yalnız bu farklılıkların hepsi sistemli bir şekilde planlanmış standartlar ile oluşmaktadır. İnternetin gelişmesindeki öncü kuruluşlardan olan IEEE, bu paketleri bir standart dâhilinde kabul etmekte ve oluşturmaktadır. Paketlerin işleme aşamaları ise TCP/IP ve OSI olmak üzere iki farklı model üzerinden yapılmaktadır. Bu modellerin farklı katmanlarında farklı protokoller çalışmaktadır. Yapılan projede gelen paketin ne olduğuna bakıp, üzerindeki protokole göre ayrıştırıp, veri tabanına kayıt yapılmaktadır. Bu projenin çıkış amacı, yerel ağ üzerinde bulunan virüslerin kendini diğer cihazlara kopyalamaya çalışması ya da ortak ağda bulunan kişilerin sisteminize erişme ya da trafiğinizi izlemek istemesi gibi durumlarda, kaynak ip adresinin ve davranışının tespit edilerek sistem yöneticisine bildirme ve kaynak ip adresinden gelecek isteklerin işlenmeden düşürülmesini sağlamaktır. Güvenlik derecesi belirlenen isteklerin güvenlik duvarı aracılığıyla engellenmesi ardından sistem yöneticisine mail yoluyla bilgi vermesi sağlanmaktadır.

## 2. MATERYAL VE METOT

### 2.1.Kullanılan yazılım Dili

Yazılım dili olarak Python tercih edilmiştir. Bunun nedeni, ilk defa bu şekilde bir proje geliştirecek olduğum için topluluk desteği ve kaynak yeterliliği iyi olmalıydı. Aynı zamanda kısıtlı sürede hızlı bir şekilde yazılımı tamamlamam gerekiyordu. Python dilini benim bu ihtiyaçlarıma yanıt verdiği için tercih ettim.

### 2.2.Kullanılan kütüphaneler

**2.2.1. Scapy:** Bu kütüphane sayesinde ağ üzerindeki trafiği dinlemekte ve ağdaki paketlerin başlık bilgilerine erişilebilmekteyim. Bu kütüphane projenin iskeletini oluşturmaktadır.



**2.2.2. Threading:** Bu modül threadları başlatmak için kullanılmaktadır. Threadlar sayesinde ağ dinleme ve veri tabanına kayıt işlemleri daha performanslı olmakta. Bu sayede paket kaçırma ihtimalini minimum seviyelere inmektedir.

**2.2.3. Time:** Bu modül mevcut zamanı öğrenmek için ve programın bellek hatası vermesini önlemek için kullanılmaktadır.

**2.2.4. Pymysql:** Bu kütüphane sayesinde python kodları ile sistemde bulunan mysql veri tabanına erişim sağlanabilmektedir.

**2.2.5. Myip:** Bu modül sayesinde programa kendi ip adresimizi söylemekteyiz. Böylece program, bizim ip adresimizde yoğun bir trafik olsa dahi ip adresimizi engellemeye çalışmayacaktır.

**2.2.6. Os:** Bu modül kod içerisinde işletim sistemi üzerinde komut yürütmeye imkan sağlamaktadır. Ayrıca dosya işlemleri için de kullanıldı.

**2.2.7. Smtplib:** Bu kütüphane, kritik olarak derecelendirilen istekleri sistem yöneticisine mail olarak göndermek için kullanılmıştır.

**2.2.8. Sys:** Bu modül hata ayıklamak için kullanılmıştır.

### 3. PROJE YAPIM AŞAMASI VE YÖNTEMİ

Bu bölümde projede yapılan yanlışlar, araştırılan konular ve kullanılan yöntemler yer almaktadır. Konulara göre bölümlendirilmiştir.

#### 3.1. Network sniffing cihaz kurulumu ve kullanımı için hazırlıklar:

İlk olarak raspberry cihaza işletim sistemi kurmayı öğrendim. Bunun için öncelikle cihazın hafıza kartını SD Card Formatter yazılımı ile formatlayıp daha sonra da win32DiskImager ile <https://www.raspberrypi.org/downloads/raspbian/> bağlantısıyla indirmiş olduğum işletim sistemini yaktım. Cihazı başlattığımda

görüntüleyebileceğim bir monitör olmadığı için kendi bilgisayarına HDMI kabloyla bağlayabilirdim ya da Ethernet kablosu ile ssh bağlantısı yaparak konsol üzerinden işlem yapabilecektim. Ama öğrendim ki bilgisayara HDMI kablosunu direkt bağladığımda hiçbir görüntü gelmemekteydi. Bunun için araştırma yaptığımda bilgisayarına VNC Viewer kurmam gerektiğini öğrendim. Daha sonra da raspberry cihazdan VNC Server'ı çalıştırmam gerektiğini öğrendim. Bunun için cihaza SSH bağlantısı sağlamam gerekiyordu. Öncelikle Ethernet kablosuyla cihazı bilgisayarına taktığımda cihazın ip adresi APIPA oldu. Bunu düzeltmek için Wi-Fi den almış olduğum interneti Ethernet portuma aktararak raspberry cihazımın bilgisayarımın vermiş olduğu bir local ip adresi aldı(örn: 192.168.54.138). İp aldırma işleminden sonra ssh bağlantısı yapmaya çalıştığımda bağlantı reddediliyordu. Bunun nedeninin raspberry cihazında ssh dosyasının olmamasından dolayı olduğunu öğrendim. Kartın içerisinde ssh.txt isimli boş bir txt dosyası oluşturduğumda ssh bağlantısını sağladım. Yalnız sonraki günlerde bilgisayarın internet trafiğini Ethernet portuna akıtmasında sıkıntı çıktı ve bağlantılarda sorun yaşamaya başladım. Bende işlemleri sanal makine üzerinde yapıp her şeyi ayarladıktan sonra raspberry cihazına yüklemeye karar verdim. Bundan sonraki çalışmalar sanal makinalar üzerinden yapılmıştır. Yazılımı yapabilmek için temel olarak python3 programlama dilini öğrendim. Daha sonra ağ trafiğini nasıl yakalayabileceğimi araştırmaya başladım. Bu zamana kadar öğrenmiş olduğum değer atamalarında, dinamik veriler üzerinde hiç işlem yapmamıştım. Bu aşamada tılandım ve bu sorunu nasıl aşabileceğimi araştırmaya başladım. Öncelikle pythonda os modülünü öğrendim. Os modülü sayesinde Python'ın, üzerinde çalışmış olduğu işletim sistemlerinde işlemler gerçekleştirebildiğini öğrendim. Os.system() komutuyla Linux konsol komutlarının burada da çalışabildiğini öğrendim. Wireshark'ın konsol sürümü olan tshark toolunu os.system("tshark") komutu ile çalıştırdığımda veriler akmaya başladı. Ama buradan daha önce de söylediğim gibi dinamik gelen verileri işleyemiyordum. Program akışı devam etmiyordu. Araştırmalarım sonucunda subprocess komutu ile geri planda ağ trafiğini dinamik olarak değişkene atayabiliyordum ama subprocess i çalıştırdığımda verimliliğinin düşük olduğunu ve işlemciyi çok yorduğunu öğrendim. Bundan dolayı bu yöntemden

vazgeçtim. Alternatif olarak hangi yöntemlerin kullanılabileceği araştırılmaya başlandı.

### 3.2.Thread araştırması, regex ile ip ve mac eşleme:

Ağ dinleme konusunda biraz yol kat etmiş durumdaydım. Yazmış olduğum kodun, hızlı bir veri akışı olduğunda yavaş kalacağını yani paket kaçırabileceği ihtimali aklıma geldi. Bu durumun önüne geçebilmek için, Thread kullanılması gerektiği öğrenildi. Threadların varlıkları bir işleme bağlı olan ve yine aynı anda birden fazla işi yapmaya yarayan yapılar olduğu öğrenildi. Ayrıca threadların diğer bir önemli özelliğinin ise bir işlemin altında çalışmasından dolayı bu işlem içerisindeki bütün haklara sahip olması olduğu öğrenildi. Bu haklardan önemli olanlardan bir tanesi de hafıza erişim hakkı olduğu öğrenildi. Bu durumda, işlem tarafından üretilen bütün işlemler, process içerisinde diğer işlemlerin kullandığı hafızaya erişebilir. Bu da paylaşımlı hafıza işlemini oluşturmakta olduğu öğrenildi. Regex kullanımıyla hızlı bir şekilde ip ve mac adreslerini tespit edebileceğimi öğrendim. Programlamada stringler üzerinde işlem yapmanın işlemciyi yorduğunu öğrendim. Regex yapısı sayesinde hızlı bir şekilde sonuç döndürülebildiğini öğrendim. Aşağıdaki regex yapıları ip adresleri ve mac adreslerini bulmak için oluşturulmuş yapılardır.

İp için: ([0-9]{1,3})\.([0-9]{1,3})\.([0-9]{1,3})\.([0-9]{1,3})

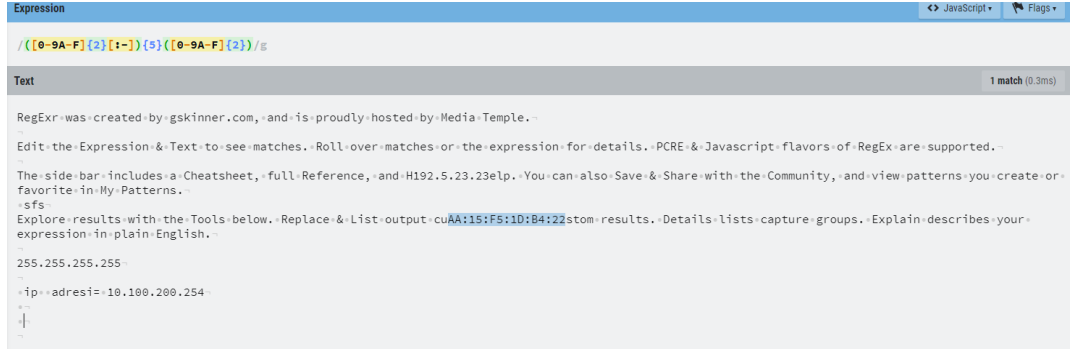
MAC için: ([0-9A-F]{2}[:-]){5}([0-9A-F]{2})

İp adresi için oluşturulan regex yapısının çıktısı şekil 1’de verilmiştir.



Şekil 1 ip regex yapısı

MAC adresi için oluşturulan regex yapısının çıktısı ise şekil 2’de verilmiştir.



Şekil 2 MAC regex yapısı

### 3.3. Python socket programlama

Yapılacak olan proje için farklı bir metod araştırılmaya başlandı. Bunun nedeni, subprocess in verimliliği azaltması ve tshark gibi farklı toolara ağ dinleme yaptırmanın verimlilik açısından olumsuz olması. Dinleme için farklı toolar yerine, direkt olarak program içerisinde dinleme işlemi için araştırma yapıldı. Socket programlama sayesinde ağ trafiği direkt olarak yakalanabilmekte olduğu öğrenildi. Projenin devamında socket programlama öğrenilerek paketler işlenecek. Socket programlamada alınan veriler ham ve işlenmemiş olarak gelmekte olduğu öğrenildi. Bu verileri işleyerek anlaşılır bir formata dönüştürme için araştırmalara başlandı. Aşağıda ham veriyi görebileceğimiz program kodu(Şekil 3) ve çıktısı(Şekil 4) verilmiştir.

```
[root@parrot]~/home/yunusemre/Desktop
#cat li.py
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
say=0
while True:
    print(s.recvfrom(65565))
    break
[root@parrot]~/home/yunusemre/Desktop
#
```

Şekil 3 Socket program kodu



```

root@kali:~# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 34
Server version: 10.1.29-MariaDB-6 Debian build-d-unstable

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create user 'yunusemre'@'localhost' identified by 'password'
-> ;
Query OK, 0 rows affected (0.00 sec)

```

Şekil 5 mysql kullanıcı oluşturma

- Daha sonra da yetkilendirme işlemi yapıldı.
  - *Grant {yetki} on {\*}.{\*} to '{kullanıcı\_adi}'@'host\_adi-ip' with grant option;*
  - Yetki alanında belirli yetkiler verilebilir ben tüm veri tabanlarına ve tablolarına erişebileceği “all privileges” yetkisini verdim. Diğer yetki tipleri Şekil 6’de verilmiştir.

Privilege	Meaning and Grantable Levels
ALL (PRIVILEGES)	Grant all privileges at specified access level except GRANT, OPTION and PROXY.
ALTER	Enable use of ALTER. Levels: Global, database, table.
ALTER ROUTINE	Enable stored routines to be altered or dropped. Levels: Global, database, routine.
CREATE	Enable database and table creation. Levels: Global, database, table.
CREATE ROLE	Enable role creation. Level: Global.
CREATE ROUTINE	Enable stored routine creation. Levels: Global, database.
CREATE TABLESPACE	Enable tablespaces and log file groups to be created, altered, or dropped. Level: Global.
CREATE TEMPORARY TABLES	Enable use of CREATE TEMPORARY TABLE. Levels: Global, database.
CREATE USER	Enable use of CREATE USER, DROP USER, RENAME USER, and REVOKE ALL PRIVILEGES. Level: Global.
CREATE VIEW	Enable views to be created or altered. Levels: Global, database, table.
DELETE	Enable use of DELETE. Level: Global, database, table.
DROP	Enable databases, tables, and views to be dropped. Levels: Global, database, table.
DROP ROLE	Enable roles to be dropped. Level: Global.
EVENT	Enable use of events for the Event Scheduler. Levels: Global, database.
EXECUTE	Enable the user to execute stored routines. Levels: Global, database, routine.
FILE	Enable the user to cause the server to read or write files. Level: Global.
GRANT OPTION	Enable privileges to be granted to or removed from other accounts. Levels: Global, database, table, routine, proxy.
INDEX	Enable indexes to be created or dropped. Levels: Global, database, table.
INSERT	Enable use of INSERT. Levels: Global, database, table, column.
LOCK TABLES	Enable use of LOCK TABLES on tables for which you have the SELECT privilege. Levels: Global, database.
PROCESS	Enable the user to see all processes with SHOW PROCESSLIST. Level: Global.
PROXY	Enable user proxying. Level: From user to user.
REFERENCES	Enable foreign key creation. Levels: Global, database, table, column.
RELOAD	Enable use of FLUSH operations. Level: Global.
REPLICATION CLIENT	Enable the user to ask where master or slave servers are. Level: Global.
REPLICATION SLAVE	Enable replication slaves to read binary log events from the master. Level: Global.
SELECT	Enable use of SELECT. Levels: Global, database, table, column.
SHOW DATABASES	Enable SHOW DATABASES to show all databases. Level: Global.
SHOW VIEW	Enable use of SHOW CREATE VIEW. Levels: Global, database, table.
SHUTDOWN	Enable use of mysqladmin shutdown. Level: Global.
SUPER	Enable use of other administrative operations such as CHANGE MASTER TO, KILL, FORCE BINARY LOGS, SET GLOBAL, and mysqladmin debug command. Level: Global.
TRIGGER	Enable trigger operations. Levels: Global, database, table.
UPDATE	Enable use of UPDATE. Levels: Global, database, table, column.
USAGE	Synonym for “no privileges”

Şekil 6 mysql yetkiler tablosu

- İlk yıldız veri tabanını belirtmekte
- İkinci yıldız tabloyu belirtmektedir
- Kullanıcı adı kısmında hangi kullanıcıya yetki ataması yapılacağı
- Host kısmında da hangi host için yetkilendirme yapılacağı belirtilmektedir. Aşağıdaki şekil 7’de yukarıda oluşturulan kullanıcıya yetki verme işleminin ekran görüntüsü bulunmaktadır.

```
MariaDB [(none)]> grant all privileges on *.* to 'yunusemre'@'localhost' with grant option;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> █
```

Şekil 7 mysql kullanıcı yetki çıktısı

- 8 *Create database {database-ismi};* diyerek bir veri tabanı oluşturdum
- 9 1. Adımda yapmış olduğum işlemi tekrar uyguladım fakat tek farkı \*.\* demek yerine {oluşturduğum\_veri\_tabanı\_ismi}.\* şeklinde belirttim.
- 10 Ve 5. Adımları tek hamlede yaptım. Use {veri tabanı\_adı}; yazdıktan sonra veri tabanının içine girdim. Daha sonra da create table {tablo\_adı} ({kolon\_adı} <özellikler>) diyerek işlemi tamamladım.

Oluşturulan veri tabanında veri tabanı ve sonrasında oluşturulan tabloda insert sorgusu çalıştırılıp kayıt işleminin başarılı olup olmadığı test edildi. Kayıt işlemi başarılıydı. Bundan sonraki çalışmalarda saldırıları tespit etmek ve nasıl davranışlarda bulunduğunu belirlemek, daha sonra da bunu kodlamak gerektiği sonucuna ulaşılmıştır.

### 3.5. Kullanılacak veri tabanı ve tabloların oluşturulması:

Tasarladığım veri tabanını aktif hale getirdim. Tabloyu oluşturmak için yazmış olduğum sorgu, şekil 8’de bulunmaktadır.

```
MariaDB (sniff)> create table db_allvalues(id bigint unsigned primary key auto increment not null, eth_mac_src varchar(20), eth_mac_dst varchar(20), eth_type int, ip_version tinyint unsigned, ip_ihl int, ip_tos int, ip_len int, ip_id int, ip_flags varchar(10), ip_frag int, ip_ttl int, ip_proto int, ip_checksum int, ip_src varchar(20), ip_dst varchar(20), icmp_type int, icmp_code int, icmp_checksum int, icmp_id int, icmp_seq int, icmp_ts_ori varchar(10), icmp_ts_rx varchar(10), icmp_ts_tx varchar(10), icmp_gw varchar(10), icmp_ptr varchar(10), icmp_reserved varchar(10), icmp_length varchar(10), icmp_addr_mask varchar(20), icmp_nexthopmtu varchar(20), icmp_unused varchar(20), tcp_sport int, tcp_dport int, tcp_seq int, tcp_ack int, tcp_dataofs int, tcp_reserved int, tcp_flags varchar(10), tcp_window int unsigned, tcp_checksum int, tcp_urgptr int, udp_sport int, udp_dport int, udp_len int unsigned, udp_checksum int, arp_hwtype int, arp_ptype int, arp_hlen int, arp_glen int, arp_op int, arp_hwsrc varchar(20), arp_psrc varchar(20), arp_hwdst varchar(20), arp_pdst varchar(20), save_time timestamp);
```

Şekil 8 mysql tablo oluşturma sorgusu

Oluşan tablo yapısının genel görünümü ise aşağıdaki şekilde(Şekil 9) verilmiştir

```
MariaDB [sniff]> describe db_allvalues;
```

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PRI	NULL	auto_increment
eth_mac_src	varchar(20)	YES		NULL	
eth_mac_dst	varchar(20)	YES		NULL	
eth_type	int(10) unsigned	YES		NULL	
ip_version	tinyint(3) unsigned	YES		NULL	
ip_ihl	int(10) unsigned	YES		NULL	
ip_tos	int(10) unsigned	YES		NULL	
ip_len	int(10) unsigned	YES		NULL	
ip_id	int(10) unsigned	YES		NULL	
ip_flags	varchar(10)	YES		NULL	
ip_frag	int(10) unsigned	YES		NULL	
ip_ttl	int(10) unsigned	YES		NULL	
ip_proto	int(10) unsigned	YES		NULL	
ip_chksum	int(10) unsigned	YES		NULL	
ip_src	varchar(20)	YES		NULL	
ip_dst	varchar(20)	YES		NULL	
icmp_type	int(10) unsigned	YES		NULL	
icmp_code	int(10) unsigned	YES		NULL	
icmp_chksum	int(10) unsigned	YES		NULL	
icmp_id	int(10) unsigned	YES		NULL	
icmp_seq	int(10) unsigned	YES		NULL	
icmp_ts_ori	varchar(10)	YES		NULL	
icmp_ts_rx	varchar(10)	YES		NULL	
icmp_ts_tx	varchar(10)	YES		NULL	
icmp_gw	varchar(10)	YES		NULL	
icmp_ptr	varchar(10)	YES		NULL	
icmp_reserved	varchar(10)	YES		NULL	
icmp_length	varchar(10)	YES		NULL	
icmp_addr_mask	varchar(20)	YES		NULL	
icmp_nextthopmtu	varchar(20)	YES		NULL	
icmp_unused	varchar(20)	YES		NULL	
tcp_spor	int(10) unsigned	YES		NULL	
tcp_dport	int(10) unsigned	YES		NULL	
tcp_seq	int(10) unsigned	YES		NULL	
tcp_ack	int(10) unsigned	YES		NULL	
tcp_dataofs	int(10) unsigned	YES		NULL	
tcp_reserved	int(10) unsigned	YES		NULL	
tcp_flags	varchar(10)	YES		NULL	
tcp_window	int(10) unsigned	YES		NULL	
tcp_chksum	int(10) unsigned	YES		NULL	
tcp_urgptr	int(10) unsigned	YES		NULL	
udp_sport	int(11)	YES		NULL	
udp_dport	int(11)	YES		NULL	
udp_len	int(10) unsigned	YES		NULL	
udp_chksum	int(10) unsigned	YES		NULL	
arp_hwtype	int(10) unsigned	YES		NULL	
arp_ptype	int(10) unsigned	YES		NULL	
arp_hwlen	int(10) unsigned	YES		NULL	
arp_plen	int(10) unsigned	YES		NULL	
arp_op	int(10) unsigned	YES		NULL	
arp_hwsrsrc	varchar(20)	YES		NULL	
arp_psrc	varchar(20)	YES		NULL	
arp_hwdst	varchar(20)	YES		NULL	
arp_pdst	varchar(20)	YES		NULL	
save_time	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

55 rows in set (0.00 sec)

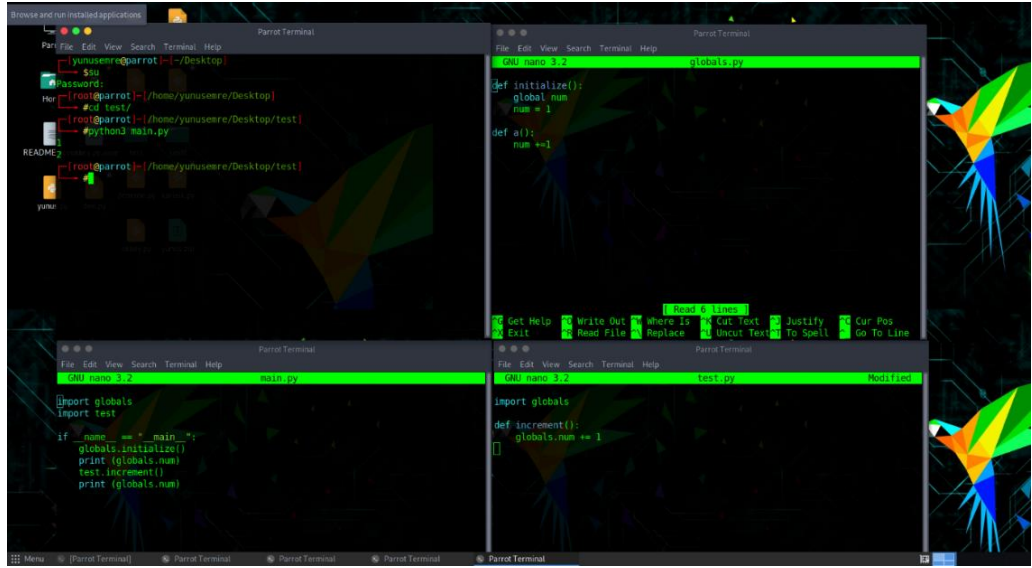
Şekil 9 veri tabanı tablosu genel görünüm

### 3.6. Global değişkenler, python kütüphane oluşturma, sqlite database:

Veri tabanı artık sorunsuz bir şekilde çalışmakta, gelen bağlantı üzerinde tanımlanmış olduğum tüm değişkenleri başarılı bir şekilde veri tabanına kaydedebilmekteyim. Artık saldırı tespit aşamasına geçmiş bulunmaktayım. Ağ dinlemek ve veri tabanına kaydetmek için yazmış olduğum kodun hız



kaybetmeden ve veri kaçırmadan işlemini yapması için saldırıyı tespit edecek kodu farklı bir python dosyasında yazmaya karar verdim. Ayrıca veri tabanına kaydetmiş olduğum verileri diğer kod dosyasında veri tabanından çekmek yerine ram üzerinde bulunan veriyi işlememin hız açısından büyük avantaj sağlayacağını araştırmalarım sonucunda öğrendim. Bundan dolayı araştırmalarımı bu konu üzerine yoğunlaştırdım. Bunu nasıl yapabileceğimi araştırırken global değişkenler tanımlayarak, ulaşmak istediğimiz değişkenin dosyasını mevcut dosyamıza import edilerek erişilebildiğini öğrendim. Bu durumu ilk olarak kavramak için test ettim. Test sonucu başarıyla sonuçlandı(Şekil 10).



Şekil 10 global değişkenlerin test edilmesi

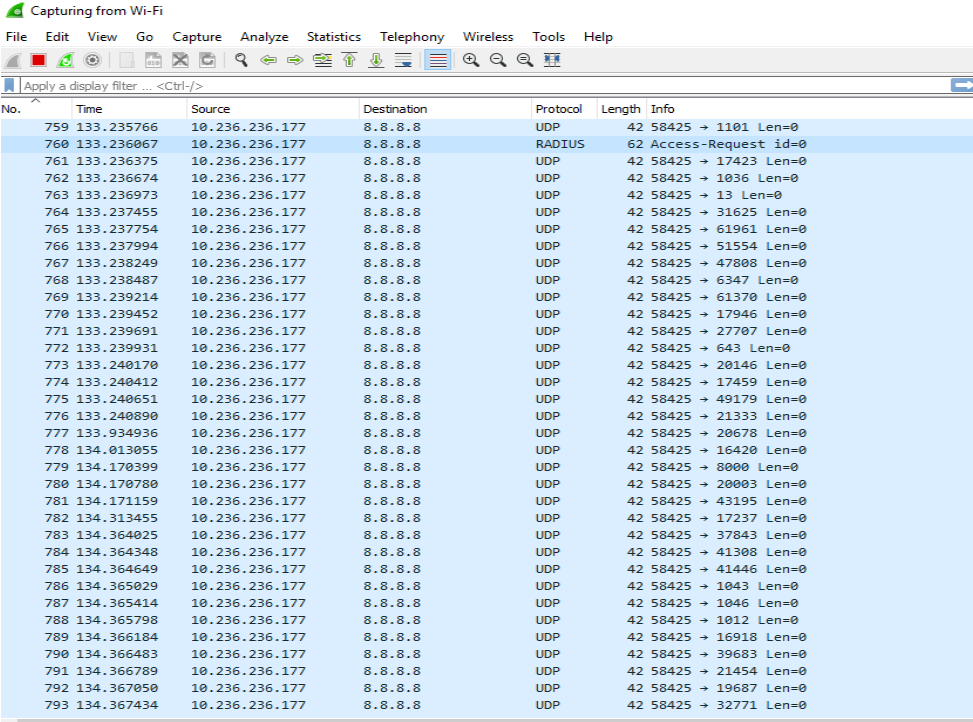
Şekil 10'u açıklayacak olursam; main.py, test.py ve globals.py adlı üç farklı python dosyam bulunmaktadır. Bunların hepsi de aynı dizin altındadırlar. İlk olarak globals.py dosyasında num isminde global bir değişken tanımladım. Bu değişken initialize fonksiyonunun içerisinde bulunmakta. Bu sayede import ettiğimiz durumda, import edilen dosyadaki fonksiyonlar çağrılabilir. Görüldüğü üzere de num değişkeninin değerini 1 olarak belirledim. A fonksiyonunun bu ekran görüntüsünde bir anlamı yoktur. Diğer testlerden kalmış, gözümünden kaçmıştır. Daha sonra test.py isminde bir python dosyası oluşturdum. Bu dosyaya da globals.py dosyasını import ettim. Burada da increment isminde

bir fonksiyon oluşturdum. Bu fonksiyon, globals içerisindeki num değişkenini 1 arttırmaktadır. Son olarak ise main.py dosyasıyla, işlemin gerçekleşip gerçekleşmediğini test edeceğim. Bunun için ilk olarak test.py ve global.py dosyalarını import ettim. Daha sonra da ilk olarak globals dosyamdaki initialize fonksiyonunu çalıştırdım. Daha sonra da num değişkeninin değerini ekrana yazdırdım. Son olarak da test.py dosyasındaki increment fonksiyonunu çağırdım. Bu fonksiyon globals.py dosyasındaki num değişkeninin değerini bir arttırmaktaydı. Bunun üzerine num değişkenini tekrar yazdırdığımda değerin 2 olduğunu gözlemledim. Bu işlemler sonucunda kendi kodumda mac adresine erişmek istediğimde başarılı olamadım. Sniff yapan dosyama saldırı tespiti yapacak dosyaya import ettiğimde sniff kodları çalışmakta, saldırı tespit edecek kodlarda akış devam etmektedir. Bunun nedenini araştırdığımda bir sonuca ulaşamadım. Bunun üzerine farklı bir yöntem araştırmaya devam ettim. Bunun üzerine sqlite veri tabanını keşfettim. Bu veri tabanı ram üzerinde oluşturulabiliyordu. Eğer bu veri tabanını global olarak tanımlayabilirsem diğer processler üzerinden de erişebilecektim. Belli bir zaman sonrasındaki verileri de sabit diske kaydedecektim. Böylelikle işlem hızım çok yüksek olacaktı. Öncelikle sqlite veri tabanının sorgulamalarını öğrenmeye çalıştım ram üzerinde testler yapmaya başladım. Yaptığım testler tek process üzerineydi. Bunu diğer processlerden de erişilebilir hale getirmeye çalıştığımda da başarılı olamadım. Veri tabanı, kendisini oluşturan processe bağlıydı. Yapılan testler başarısızlıkla sonuçlandı ve zamanın az kalmasından dolayı mevcut durumda kullanmış olduğum veri tabanını kullanmaya devam etmeye, saldırı tespiti yapacak process dosyasının da veri tabanı sorgusu yaparak işlemlerine devam etmesi konusunda karara vardım.

### **3.7. Wireshark ile UDP paket içeriğini analiz etme ve Nmap UDP tarama:**

Veri tabanı işlemleri hazır olduktan sonra yeni bir python dosyası oluşturuldu. Bu dosyanın amacı, kendisine tanımlanan saldırı tiplerini tespit ederek bildirimde bulunmak. Öncelikle UDP taramayı tespit edecek kodu yazacaktım. UDP tarama hakkında; SYN, ACK, RST paketleri bulunmamakta olduğu öğrenildi. O yüzden

nmap, basitçe hedef portlara, UDP paketleri gönderiyor. Eğer o portta bir servis yok ise, ICMP Port Unreachable mesajı geliyor. Bu da hedef portunun kapalı olduğu anlamına geliyor. Hiç birşey gelmezse, açık olan bir port olduğu anlamına geldiği öğrenildi. 2 olasılık olduğu için bu durumda false positive durumuda olabileceği öğrenildi. False positive gelmesinin sebebi firewall tarafından paket düşürülmüş olabileceği öğrenildi. Bundan dolayı yanlış yere bir portun açık olduğu çıkarımı da yapılabileceği öğrenildi. Yalnız bunun için UDP taramada sistemin nasıl davrandığını incelemek gerekiyordu. Öncelikle nmap aracını kullandım. Nmap aracı ile taramanın yapısını çözmek amacıyla, Google DNS sunucularına UDP tarama başlattım (Şekil 11).

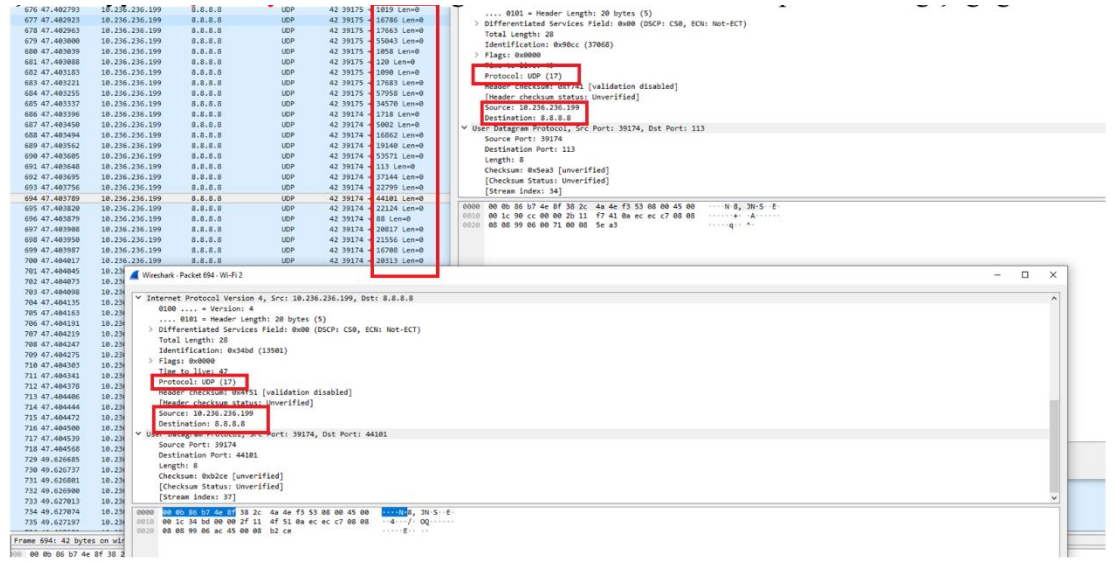


No.	Time	Source	Destination	Protocol	Length	Info
759	133.235766	10.236.236.177	8.8.8.8	UDP	42	58425 → 1101 Len=0
760	133.236067	10.236.236.177	8.8.8.8	RADIUS	62	Access-Request id=0
761	133.236375	10.236.236.177	8.8.8.8	UDP	42	58425 → 17423 Len=0
762	133.236674	10.236.236.177	8.8.8.8	UDP	42	58425 → 1036 Len=0
763	133.236973	10.236.236.177	8.8.8.8	UDP	42	58425 → 13 Len=0
764	133.237455	10.236.236.177	8.8.8.8	UDP	42	58425 → 31625 Len=0
765	133.237754	10.236.236.177	8.8.8.8	UDP	42	58425 → 61961 Len=0
766	133.237994	10.236.236.177	8.8.8.8	UDP	42	58425 → 51554 Len=0
767	133.238249	10.236.236.177	8.8.8.8	UDP	42	58425 → 47808 Len=0
768	133.238487	10.236.236.177	8.8.8.8	UDP	42	58425 → 6347 Len=0
769	133.239214	10.236.236.177	8.8.8.8	UDP	42	58425 → 61370 Len=0
770	133.239452	10.236.236.177	8.8.8.8	UDP	42	58425 → 17946 Len=0
771	133.239691	10.236.236.177	8.8.8.8	UDP	42	58425 → 27707 Len=0
772	133.239931	10.236.236.177	8.8.8.8	UDP	42	58425 → 643 Len=0
773	133.240170	10.236.236.177	8.8.8.8	UDP	42	58425 → 20146 Len=0
774	133.240412	10.236.236.177	8.8.8.8	UDP	42	58425 → 17459 Len=0
775	133.240651	10.236.236.177	8.8.8.8	UDP	42	58425 → 49179 Len=0
776	133.240890	10.236.236.177	8.8.8.8	UDP	42	58425 → 21333 Len=0
777	133.934936	10.236.236.177	8.8.8.8	UDP	42	58425 → 20678 Len=0
778	134.013055	10.236.236.177	8.8.8.8	UDP	42	58425 → 16420 Len=0
779	134.170399	10.236.236.177	8.8.8.8	UDP	42	58425 → 8000 Len=0
780	134.170780	10.236.236.177	8.8.8.8	UDP	42	58425 → 20003 Len=0
781	134.171159	10.236.236.177	8.8.8.8	UDP	42	58425 → 43195 Len=0
782	134.313455	10.236.236.177	8.8.8.8	UDP	42	58425 → 17237 Len=0
783	134.364025	10.236.236.177	8.8.8.8	UDP	42	58425 → 37843 Len=0
784	134.364348	10.236.236.177	8.8.8.8	UDP	42	58425 → 41308 Len=0
785	134.364649	10.236.236.177	8.8.8.8	UDP	42	58425 → 41446 Len=0
786	134.365029	10.236.236.177	8.8.8.8	UDP	42	58425 → 1043 Len=0
787	134.365414	10.236.236.177	8.8.8.8	UDP	42	58425 → 1046 Len=0
788	134.365798	10.236.236.177	8.8.8.8	UDP	42	58425 → 1012 Len=0
789	134.366184	10.236.236.177	8.8.8.8	UDP	42	58425 → 16918 Len=0
790	134.366483	10.236.236.177	8.8.8.8	UDP	42	58425 → 39683 Len=0
791	134.366789	10.236.236.177	8.8.8.8	UDP	42	58425 → 21454 Len=0
792	134.367050	10.236.236.177	8.8.8.8	UDP	42	58425 → 19687 Len=0
793	134.367434	10.236.236.177	8.8.8.8	UDP	42	58425 → 32771 Len=0

Şekil 11 google dns UDP taraması

Tarama sonucunda oluşan paketleri wireshark ile açarak içeriklerini inceledim. IP başlığında protokolün 17(UDP) olduğu görülmüştür. Farklı portlara normalden daha fazla istek gitmişse bunun UDP taraması olduğu anlaşılabilir. Açılan UDP paket içerikleri aşağıdaki şekilde verilmiştir.

Şekilde 12’de işaretli yerler incelendiğinde devamlı olarak hedef portların değiştiği gözlemlenmiştir.



Şekil 12 UDP tarama sonucunun incelenmesi

### 3.8. UDP port tarama eylemini tespit edecek fonksiyonun yazılması, MITM saldırısının araştırılması:

Yazacağım kodda UDP port taramasının olup olmadığını belirlemem gerekiyordu. Zaten veri tabanında saldırganın bekletme işlemi de kullanacağını düşünerek, veri tabanına kaydedilen trafikteki son 10 saniyeye göre işlem yapmaya karar verdim. Tarama tespiti yapacak olan kod aşağıdaki şekildedir(Şekil 13).

```
db=mysql.connect(user='yunus', passwd='yunus', host='localhost', db='sniff')
cr=db.cursor()

def udp_tarama():
    cr.execute('select ip.src, ip.dst, count(*) from (select ip.src, ip.dst, count(*) from db.allvalues where save time > (now() - interval 10 second) and udp.dport <> "NULL" group by ip.src, ip.dst, udp.dport) as tablo group by ip.src, ip.dst')
    rows=cr.fetchall()
    for row in rows:
        if (row[2] > 10):
            print("(! ip adresinden, (! ip adresinin (! portuna UDP taraması yapılmıştır.\n(! ip adresi Güvenlik Derecesi 3 olarak kara listeye ve log dosyasına eklenmiştir.".format(row[0], row[1], row[2], row[3]))
            query='insert into blacklist(ip) values(%s)'.format(row[0])
            cr.execute(query, row[0])
            db.commit()
        elif (row[2] > 5):
            print("(! ip adresinden, (! ip adresinin (! portuna UDP taraması yapılmıştır.\n(! ip adresi Güvenlik Derecesi 2 olarak loglanmıştır.".format(row[0], row[1], row[2], row[3]))
        elif (row[2] > 2):
            print("(! ip adresinden, (! ip adresinin (! portuna UDP taraması yapılmıştır.\n(! ip adresi Güvenlik Derecesi 1 olarak loglanmıştır.".format(row[0], row[1], row[2], row[3]))
```

Şekil 13 UDP taramayı tespit eden kod

Bu kodu açıklamak gerekirse ilk satırda veri tabanı bağlantısı oluşturuluyor. Daha sonra da veri tabanında işlem yapabilmek için bir cursor oluşturmak gerekiyordu. cr isminde bir cursor oluşturarak işleme başladım. Öncelikle bir

fonksiyon oluřturdum. Bu fonksiyonun grevi yerel ađdaki UDP taramalarını tespit etmektir. Cr.execute komutuyla veri tabanında komut alıřtıracađımızı programa belirtiyoruz. Komutumuz bir sorgulama komutudur. Sorgu komutu sayesinde bir ip adresinden bařka bir ip adresine yapılan UDP isteklerinde ka farklı port hedef alındıđı ıkartılıyor. Bu sayede hangi ip adresinin hangi ip adresine UDP taraması yaptıđı belirlenebiliyor. Yapılan iřlemler daha sonra bir txt dosyasına log olarak kaydedilecektir. Kodda belirli bir ip adresinden bařka bir ip adresine 10’dan fazla tarama yapmıřsa bu kritik olarak deđerlendirilmekte ve blacklist adlı tabloya tarama yapan ip adresi ekleniyor. 10 ile 5 arasındaysa sistem gvenlik derecesi 2 olarak uyarı veriyor. Eđer port sayısı 5 ile 2 arasındaysa gvenlik derecesi 1 olarak log kaydediliyor. Port sayısı 2 ya da daha azsa hibir aksiyon almadan geiliyor.

Mitm: Bu iřlem iin ncelikle MITM saldırısı yapılacađı zaman ađ paketlerinde nasıl bir deđerim olduđu gzlemlenmeye alıřılmıřtır. Bunun iin ađ dinleyen kodumuz alıřtırılıp veri tabanından sorgular ile hangi deđerlerde deđerim olduđu gzlemlenmeye alıřılmıřtır. Saldırıyı yapmak iin, Debian dađıtımlarından olan Parrot OS sistemindeki “websploit” toolu kullanılmıřtır. Bunun iin komut satırına websploit diyerek toolu bařlattım. Daha sonra help komutu ile neler yapabileceđimi grdm(řekil 14).



```

wsf > show modules
-----
Web Modules
-----
web/apache_users      Scan Directory Of Apache Users
web/dir_scanner        Directory Scanner
web/wmap              Information Gathering From Victim Web Using (Metasploit Wmap)
web/pma               PHPMyAdmin Login Page Scanner
web/cloudflare_resolver CloudFlare Resolver

Network Modules
-----
network/arp_dos        ARP Cache Denial Of Service Attack
network/mfod           Middle Finger Of Doom Attack
network/mitm           Man In The Middle Attack
network/mlitm          Man Left In The Middle Attack
network/webkiller       TCP Kill Attack
network/fakeupdate      Fake Update Attack Using DNS Spoof
network/arp_poisoner    ARP Poisoner
exploit/autopwn         Metasploit Autopwn Service
exploit/browser_autopwn Metasploit Browser Autopwn Service
exploit/java_applet     Java Applet Attack (Using HTML)
wireless/wifi_jammer    Wifi Jammer
wireless/wifi_dos       Wifi Dos Attack
wireless/wifi_honeypot  Wireless Honeypot(Fake AP)
wireless/mass_deauth     Mass Deauthentication Attack
bluetooth/bluetooth_pod Bluetooth Ping Of Death Attack

```

Şekil 15 websploit modüller çıktısı

Çıktı sonucunda “Network Modules” modüllerinden olan netwok/mitm modulu use komutuyla seçilmiştir(Şekil 16). Daha sonra da Show options diyerek hangi ayarları değiştirmem gerektiğini gördüm.



```

wsf > use network/mitm
wsf:MITM > show options
Options      Value      Description
-----
Interface    eth0      Network Interface Name
ROUTER       192.168.1.1 Router IP Address
TARGET mitm() 192.168.1.2 Target IP Address
SNIFFER orgu bos driftnet Sniffer Name (Select From Sniffer List)
SSL (wall: 80) true orsa kod calissin yes SSLStrip, For SSL Hijacking(true or false)

Sniffers
MyLoca Description
-----
dsniiff mitm Sniff All Passwords
msgsnarf mitmip Sniff All Text Of Victim Messengers
urlsnarf destin Sniff Victim Links
driftnet destin Sniff Victim Images

for row in rows:
    print "Source IP Address: %s" % row["ip_src"]

wsf:MITM >

```

Şekil 16 websploit options çıktısı

Set komutu kullanılarak Router ve Target değerleri değiştirilmiştir. Run diyerek de saldırı gerçekleştirilmiştir. Aşağıdaki ekran görüntüsünde(Şekil 17) veri tabanına kaydedilen bir trafik ile ilgili ICMP kayıtlarından tip ve kodu, kaynak ve hedef ip adresleri, kaynak ve hedef mac adres bilgisi, ICMP gateway adres bilgileri toplanmıştır.

```

MariaDB [sniff]> select icmp_type, icmp_code, icmp_gw, eth_mac_dst, eth_mac_src, ip_src, ip_dst from db_allvalues where icmp_type <> "NULL" and icmp_type <= 3;
+-----+-----+-----+-----+-----+-----+-----+
| icmp_type | icmp_code | icmp_gw | eth_mac_dst | eth_mac_src | ip_src | ip_dst |
+-----+-----+-----+-----+-----+-----+-----+
| 8 | 0 | NULL | 00:50:56:fe:56:b4 | 00:0c:29:be:5a:76 | 192.168.64.149 | 216.239.38.120 |
| 8 | 0 | NULL | 00:50:56:fe:56:b4 | 00:0c:29:be:5a:76 | 192.168.64.149 | 216.239.38.120 |
| 8 | 0 | NULL | 00:50:56:fe:56:b4 | 00:0c:29:be:5a:76 | 192.168.64.149 | 216.239.38.120 |
| 8 | 0 | NULL | ff:ff:ff:ff:ff:ff | 00:50:56:e4:71:44 | 192.168.64.254 | 192.168.64.144 |
| 8 | 0 | NULL | 00:0c:29:44:8b:b6 | 00:50:56:fe:56:b4 | 192.168.64.254 | 192.168.64.144 |
| 5 | 1 | 192.168.64 | 00:0c:29:f3:ee:7e | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.151 |
| 5 | 1 | 192.168.64 | 00:0c:29:f3:ee:7e | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.151 |
| 5 | 1 | 192.168.64 | 00:0c:29:f3:ee:7e | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.151 |
| 5 | 1 | 192.168.64 | 00:0c:29:f3:ee:7e | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.151 |
| 5 | 1 | 192.168.64 | 00:0c:29:f3:ee:7e | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.151 |
| 5 | 1 | 192.168.64 | 00:50:56:fe:56:b4 | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.2 |
| 5 | 1 | 192.168.64 | 00:50:56:fe:56:b4 | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.2 |
| 5 | 1 | 192.168.64 | 00:50:56:fe:56:b4 | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.2 |
| 5 | 1 | 192.168.64 | 00:50:56:fe:56:b4 | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.2 |
| 5 | 1 | 192.168.64 | 00:0c:29:f3:ee:7e | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.151 |
| 5 | 1 | 192.168.64 | 00:50:56:fe:56:b4 | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.2 |
| 5 | 1 | 192.168.64.2 | 00:0c:29:f3:ee:7e | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.151 |
| 5 | 1 | 192.168.64.2 | 00:0c:29:f3:ee:7e | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.151 |
| 5 | 1 | 192.168.64.2 | 00:0c:29:f3:ee:7e | 00:0c:29:be:5a:76 | 192.168.64.150 | 192.168.64.151 |
+-----+-----+-----+-----+-----+-----+-----+
20 rows in set, 1 warning (0.11 sec)

```

Şekil 17 mitm saldırı sonrası veri tabanı kayıtları

Bu sorgu yalınlaştırılmış bir sorgudur. MITM yapılan bir ağdaki değişimler gözlenebilmektedir. Saldırı işleminden sonra icmp\_gw gateway adresinin ip adresini üzerinde bulundurmakta fakat normal bir trafikte NULL değerini almaktaydı. Ayrıca MITM başladıktan sonra icmp tip ve kod değerleri değişmişti. Bu değişken içeriklerine göre MITM tespiti yapabileceğimi düşünerek aşağıdaki(Şekil 18) python fonksiyonunu yazdım.



```

def mitm():
    sorgu_bosmu=cr.execute('select icmp_gw, ip_src, ip_dst from db_allvalues where save_time > (now() - interval 10 second) and icmp_code = 1 and icmp_type = 5') #burasi if blojunda
    olmalı. Bos dönmüyorsa kod çalışsın
    if (sorgu_bosmu != 0):
        rows=cr.fetchall()
        MyLocalIP = myip.myip()
        cr.execute('select ip_mitm from blacklist_mitm')
        mitm_rows = cr.fetchall()
        mitmip=""
        destination_ip=""
        destination_gw=""
        for row in rows: #disardan yakalanan trafik sorgusunda ortadaki adamin ip adresi, source ip adresindeki, kendi ip adresimiz olmayan ip adresidir.(bu cümleyi tek okumada anlayan
            dahidir :0)
            if (row[1] != MyLocalIP):
                mitmip=row[1]
                destination_ip=row[2]
                destination_gw=row[0]
                break
            kntrl = True
            for mitm_row in mitm_rows:
                if (mitm_row[0] == mitmip):
                    kntrl = False
                    break
            if kntrl: #mitm tablosunda tespit ettiğimiz ip adresi yok demek
                print ("{} adresine sahip bilgisayar, {} ip adresli bilgisayara, {} gateway adresi için mitm saldırısı yapıyor olabilir.\n{} ip adresi blacklist_mitm tablosuna kaydedilmiştir.
                \nİlgili IP, Sniff isteminin firewall block ip tablosuna da eklenmiştir.".format(mitmip, destination_ip, destination_gw, mitmip))
                cr.execute("insert into blacklist_mitm (ip_mitm) values({})".format(mitmip))
                db.commit()
                query="iptables -A INPUT {} -j DROP".format(mitmip)
                os.system(query)
            else:
                print ("{} adresine sahip bilgisayar, {} ip adresli bilgisayara, {} gateway adresi için mitm saldırısı yapıyor olabilir.\n{} ip adresi Local sistemde blacklist_mitm tablosuna
                daha önceden kaydedilmişti.".format(mitmip, destination_ip, destination_gw, mitmip))
#56

```

Şekil 18 mitim saldırı tespit kodu

Veri tabanından icmp tipi 5, kodu 1 olan ve son 10 saniye içerisinde kaydedilmiş olan trafik bilgileri çekilmekte. Eğer bu işlem sonucu boş dönmüyorsa bir eylem yakaladığı belli oluyor ve log dosyalarına da bir MITM saldırısı gerçekleşiyor olabileceği belirtiliyor. Daha sonra da saldırgan olduğu tespit edilen ip adresi Linux içerisindeki güvenlik duvarına kaydediliyor. Bu sayede saldırgan tarafından MITM saldırısına uğrama olasılığı azalıyor. Ayrıca saldırgan ip blacklist\_mitm tablosuna kaydediliyor. Bu sayede ağ üzerinde şüpheli işlemlerde bulunan ip listesi elde edilmiş oluyor.

### 3.9. Python SMTP mail gönderme:

Yazmış olduğum kodlar artık düzgün bir şekilde gelen paketleri açıp istediğim bilgilerini çekiyor ve veri tabanına kaydediyor. Ayrıca saldırı olup olmadığını algılayacak olan kod ise veri tabanına kaydedilen bilgileri işleyerek saldırı olup olmadığını algılayabiliyor. Bundan sonraki yaptığım çalışmalarda ise herhangi bir saldırı durumunda yöneticinin de haberdar olması. Bunun için bildiğim iki yöntemi de kullanacağım. Birincisi yöneticiye kritik olarak belirlenen eylemleri e-posta ile bildirmesi ve az da olsa şüpheli durumları log dosyasına kaydetmek. Log dosyalarını da gerçek sistemlerdeki ajanlar, siem merkezine gönderecektir. İlk yaptığım çalışma mail gönderme çalışması. Bunun için bir mail fonksiyonu oluşturdum(Şekil 19).

```

def mail(subject, alert):
    message= MIMEMultipart()

    message["From"] = "bitirme.projesi@toybelgi.com" #Mail'i gönderen kişi
    message["To"] = "yed05151@gmail.com" #Mail'i alan kişi
    message["Subject"] = subject #Mail'in konusu

    body= alert #Mail içerisinde yazacak içerik

    body_text = MIMEText(body,"plain") #
    message.attach(body_text)

    #Gmail serverlerine bağlanma işlemi.
    try:
        mail = smtplib.SMTP("mail.toybelgi.com",587)

        mail.ehlo()

        mail.starttls()

        mail.login("bitirme.projesi@toybelgi.com","BitirmeProjesi1")

        mail.sendmail(message["From"],message["To"],message.as_string())

        print("Mail Başarılı bir şekilde gönderildi.")

        mail.close()
    except:
        #Eğer mesaj gönderirken hata ile karşılaşırsak except çalışır.
        sys.stderr.write("Bir hata oluştu. Tekrar deneyin...")
        sys.stderr.flush()

```

Şekil 19 sistem yöneticisine mail gönderen python fonksiyonu

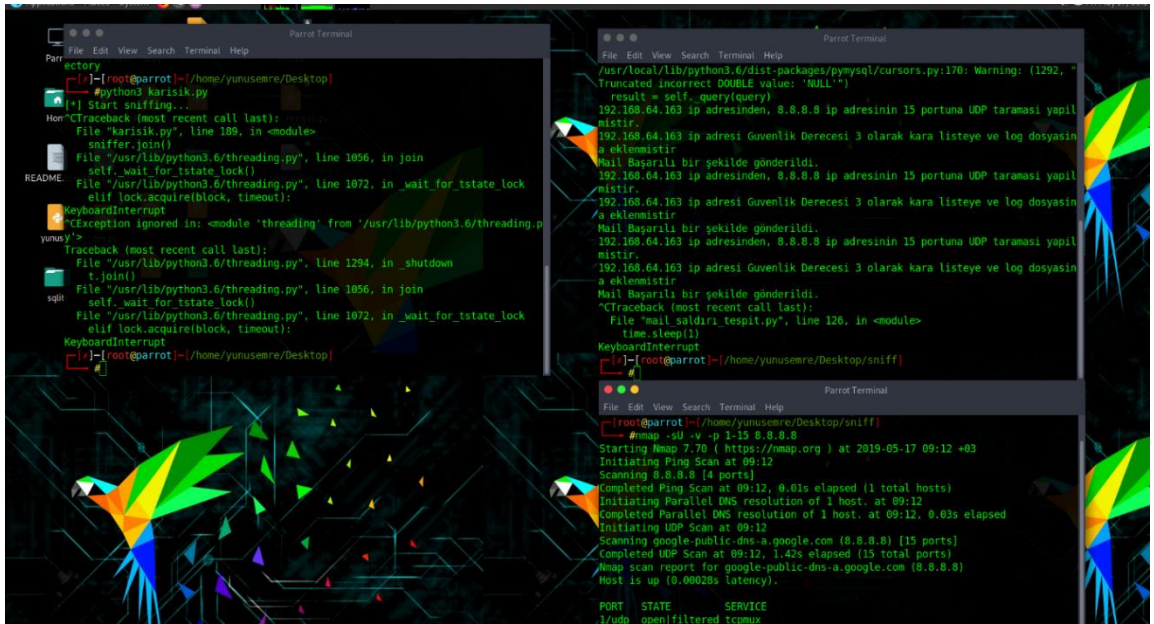
Bu fonksiyon iki değer alıyor. Bu değerlerden subject parametresi, hangi saldırı tipine ait olduğunu belirlememi sağlıyor. Alert ise içeriğin ayrıntılarını bildiriyor. Alert girdisi Şekil 20’de message değişkenidir.

```
def udp_tarama():
    cr.execute('select ip_src, ip_dst, count(*) from (select ip_src, ip_dst, count(*) from db_allvalues where
    ip_src, ip_dst, udp_dport) as tablo group by ip_src, ip_dst')
    rows=cr.fetchall()
    for row in rows:
        if (row[2] > 10):
            message = ("{} ip adresinden, {} ip adresinin {} portuna UDP taramasi yapilmistir.\n{} ip adresi
            eklenmistir".format(row[0], row[1], row[2], row[0]))
            print(message)

            log_file(message, "\n\nKritik\t")
            subject = "UDP taraması"
            query="insert into blacklist(ip) values(%s)"
            mail(subject, message)
            cr.execute(query, row[0])
            db.commit()
```

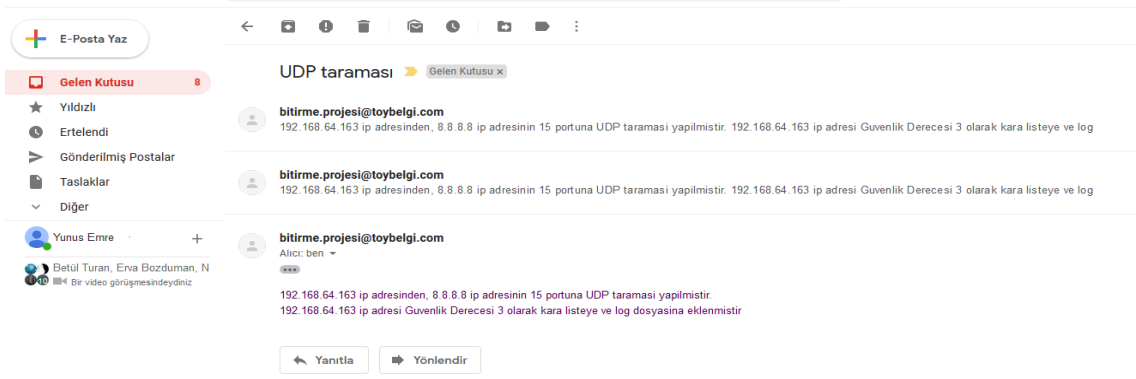
Şekil 20 alert mesajını oluşturan message değişkeni

Kodlarımı çalıştırıp 8.8.8.8 ip adresinin ilk 15 portuna nmap aracı ile UDP taraması gerçekleştirdim. Bunun sonucunda görüldüğü üzere(Şekil 21) mail gönderilmiştir.



Şekil 21 mailin gönderildiğine dair ekran görüntüsü

Gelen mail ise şekil 22’de ekran görüntüsünde verilmiştir.



Şekil 22 mailin gelen kutusundaki ekran görüntüsü

### 3.10. Log dosyası oluşturma:

Güvenlik alertlerini bir log dosyasına kaydetmek gerekiyordu. Çünkü veri tabanındaki değerler, belirlenen bir sınır günden sonra silinmesi gerekiyor. Log dosyaları ise daha uzun süre saklanabiliyor. Çünkü daha az boyut kaplamakta ve gereksiz bilgiler bulundurmamaktadır. Log dosyasını da aşağıdaki şekilde (Şekil 23) olduğu gibi oluşturup kayıt işlemini yapmaktayım.

```
def log_file(message, alert_type):  
  
    with open('log_file.log', 'a') as f:  
        f.write(alert_type + message + "\t" + time.ctime())  
        f.flush()
```

Şekil 23 log dosyasını oluşturan python fonksiyonu

Bu fonksiyon da iki adet girdi parametresi almaktadır. Değerleri de mail fonksiyonunda olduğu gibidir. With open ile dosyayı açıyorum. With open kullanmamın sebebi hata oluştuğunda dosyayı düzgün bir şekilde kapatarak bozulmaları önlemekte işlem bittikten sonra da dosyayı kapatmaktadır. İlk tırnaklar arasındaki değer dosyanın ismini, ikinci değer ise açılma modunu belirtmektedir. A modu dosyaya ekleme yaparak yazmakta ve dosya yoksa oluşturmaktadır. W modu ile açtığımda ilk önce kaydedilmiş değer

silinmekteydi. f.flush() fonksiyonu ise yazılacak verileri bellekte bekletmeden hemen yazma işlemini gerçekleştiriyor. Bu sayede bellek tüketilmemiş oluyor ve anlık olarak takip işlemi yapılabiliyor.

#### **4. SONUÇLAR:**

- Proje sonucunda UDP port tarama tespit edilebilmektedir.
- Ortadaki adam saldırısı(Mitm) tespit edilebilmektedir.
- Lokal ağdaki tüm trafik izlenebilmekte ve veri tabanına kaydedilebilmekte.
- Herhangi bir güvenlik derecesine giren trafik log dosyasına kaydedilmekte.
- Kritik seviyedeki tarama ya da saldırı, yöneticiye mail ile bildirilmekte.
- Saldırgan belirtiler gösteren ip adresinden gelen paketler düşürülmektedir.

#### **5. PROJENİN GELİŞTİRİLMESİ İÇİN ÖNERİLER:**

- Paketleri yakalayan ve pars eden özel bir kütüphane geliştirilebilir.
- Threadlar daha verimli bir şekilde yapılandırılabilir.
- Saldırı tespiti için kullanılan girdiler kabul görmüş standartlara göre işlenerek saldırı tespitleri gerçekleştirilebilir. Bu şekilde False Positive olma olasılığı düşürülür.
- Yerel ağda bulunan diğer istemciler ve sunuculara saldırgan IP adresini engellemesi için ajan geliştirilerek kurulum yapılabilir.
- Farklı saldırı tiplerinin tespiti için kod genişletildiğinde veri tabanı yetersiz kalabilir. Bunun için optimizasyon ya da veri tabanı değişikliği yapılabilir.

## 6. KAYNAKLAR:

Aşağıda verilen kaynaklar taranmış ve proje için yararlı bilgiler için kullanılmıştır.

- <https://www.algoritmauzmani.com/raspberry-pi-dersleri/raspberry-pi-ssh-aktif-etme/>
- <https://www.raspi-tr.com/2014/10/12/raspberry-piye-ag-kablosu-ile-dogrudan-baglanma/>
- <https://www.sdcard.org/downloads/formatter/>
- <https://sourceforge.net/projects/win32diskimager/>
- <https://www.algoritmauzmani.com/raspberry-pi-dersleri/raspberry-pi-3-ethernet-boot-network-boot/>
- <https://maker.robotistan.com/raspberry-pi-dersleri-3-uzaktan-baglanti-yontemleri/>
- <http://www.ferzendervarli.com/raspberry-pi-ssh-baglanma-ve-vnc-server-kurulumu/>
- <http://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/06/tcpdump-kullan%C4%B1m%C4%B1>
- <https://pentesterest.wordpress.com/2018/01/28/scapy-nedir-ne-ise-yarar-paket-gonderme-paket-yakalama/>
- <http://ysar.net/python/soket-socket.html>
- <https://danielmiessler.com/study/tcpdump/>
- <https://canyoupwn.me/tcpdump-ile-network-trafik-analizi/>
- <https://hackertarget.com/tshark-tutorial-and-filter-examples/>
- <https://realpython.com/python-sockets/>
- <https://amits-notes.readthedocs.io/en/latest/networking/tcpdump.html>
- <http://www.devshed.com/c/a/Python/Sockets-in-Python-Into-the-World-of-Python-Network-Programming/>
- <https://www.slideshare.net/bgasecurity/tshark>
- <https://github.com/birolemekli/network-packet-sniffer>
- <https://stackoverflow.com/questions/55748575/network-sniffing-raw-format-with-socket-programming-at-python>
- <https://github.com/besimaltnok/scapy-cheatsheet>

- <https://www.wireshark.org/docs/man-pages/tshark.html>
- <http://blog.lifeoverip.net/2010/11/21/tcpip-aglarda-ileri-seviye-paket-analizi-tshark/>
- <https://github.com/wireshark/wireshark>
- <http://uygulamalar.blogcu.com/python-ile-soket-programlama/4039294>
- <https://www.youtube.com/watch?v=6jteAOmdsYg&list=PLhTjy8cBISerYuLZUvVOYsR1giva2payF>
- <https://www.slideshare.net/bgasecurity/tshark>
- <https://belgeler.yazbel.com/python-istihza/>
- <https://medium.com/python/python-os-mod%C3%BCI%C3%BC-a699a681c734>
- <https://wiki.python.org/moin/AdvancedBooks?action=fullsearch&context=180&value=system&titlesearch=Titles>
- <https://snippplr.com/view/3579/live-packet-capture-in-python-with-pcap/>
- [https://docs.python.org/3/library/struct.html#struct.unpack\\_from](https://docs.python.org/3/library/struct.html#struct.unpack_from)
- <https://www.wireshark.org/docs/dfref/>
- <https://www.wireshark.org/docs/dfref/m/mysql.html>
- <https://github.com/wil3/scapy-standalone/blob/master/scapy.py>
- <http://www.bitforestinfo.com/2017/01/how-to-write-simple-packet-sniffer.html>
- <http://www.offensiveops.io/tools/byos-build-your-own-sniffer-python-tutorial/>
- [http://www.belgeler.org/bgnet/bgnet\\_theory-lowlevel.html](http://www.belgeler.org/bgnet/bgnet_theory-lowlevel.html)
- [https://docs.python.org/3/library/struct.html#struct.unpack\\_from](https://docs.python.org/3/library/struct.html#struct.unpack_from)
- [https://www.colasoft.com/help/7.1/appe\\_codes\\_ethernet.html](https://www.colasoft.com/help/7.1/appe_codes_ethernet.html)
- <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>
- <https://ibrahimirdem.blogspot.com/2015/06/python-ile-mysql-baglanti-mysqlldb.html>
- <https://medium.com/@oguzalbast02/ortadaki-adam-sald%C4%B1r%C4%B1s%C4%B1-mitm-detay%C4%B1-anlat%C4%B1m-5e5f86af1d6a>
- <https://ertugruldiniz.com/python-smtp-modulu-ile-mail-gonderme-151>
- <https://emregeldegul.net/2017/08/python-smtplib-modulu-ile-e-posta-gonderme/>
- <https://medium.com/@bkokkus/iptables-2538de7e93ec>
- <https://gist.github.com/spinpx/263a2ed86f974a55d35cf6c3a2541dc2>

- <https://gist.github.com/zbetcheckin/617a380fe82fe86f1f3add27cfb48513>

## **7. EKLER:**

Dokümanda anlatılan programın kaynak kodları ve programın çalıştırılarak çıktılarının anlatıldığı videolar zip dosyası halinde ekte sunulmuştur.

**7.1.** Saldiritespit.py

**7.2.** Sniffing.py

**7.3.** Smtplib.py

**7.4.** Myip.py

**7.5.** Vtyedek.sql