KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ

MÜHENDİSLİK MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

2020-2021 GÜZ DÖNEMİ

BİL 409

MAKİNE ÖĞRENMESİ DERSİ

FINAL PROJESİ

BÜYÜK VERİDE SALDIRI TESPİT SİSTEMLERİ UYGULAMA RAPORU

Hazırlayan

EBRU ŞİMŞEK

18110131003 (ebrusimsek201@gmail.com)

Ders Öğretim Üyesi

Dr. Öğr. Üyesi YAVUZ CANBAY (yavuzcanbay@ksu.edu.tr)

Özet__Bu çalışmada, büyük veride ağ saldırısı tespiti yapan bir sınıflandırıcı oluşturmak için KDD Cup 1999 veri seti, akan büyük veri kümeleri ile çalışmayı sağlayan Apache Spark, makine öğrenimi kütüphanesi MLlib ve python ile spark kullanımı sağlayan pyspark kullanılmıştır.

Anahtar kelimeler__Apache Spark, MLlib, büyük veride saldırı tespiti, pyspark

GİRİŞ

Günümüzde sürekli online olduğumuz için veri üretmekteyiz. Üretilen veriler büyük veri analitiğinde işlenerek kullanıcı davranışlarına yönelik çözümler sunulmaktadır. Ancak bu büyük verinin işlenmesi önemli olduğu kadar dışarıdan gelebilecek saldırılardan korunmasıda önemlidir. Verinin gizliliğini, bütünlüğünü, erişilebilirliğini tehlikeye sokan bu saldırıların türlü çeşidi vardır. Denial of Service (DoS), User to root (U2R), Remote to Local (R2L) ve Probing saldırı çeşitlerini oluşturur.

DoS saldırıları ağ üzerinde TCP/IP açıklarını hedef alarak sunulan hizmeti engeller.U2R sistem açıkları kullanılarak kullanıcı hesabının yönetici hesabına çevrilmesi ile yönetim yetkisinin ele geçirilmesidir.R2L hedefteki bilgisayara paket gönderip uzaktan erişerek kullanıcı yetkisi kazanmaktır. Probing ise hedef makinenin IP adresi, portlar, ağ, ağdaki bilgisayarlar, kullanıcı sayısı, işletim sistemi gibi gizli bilgilerini tarama saldırısıdır.

Saldırıları izlemek için saldırı tespit sistemleri geliştirilmektedir. Büyük veride sürekli veri akışı olduğu için gerçek zamanlı tespit yapabilmek önemlidir. Böylece anında rapor oluşturularak yetkili uyarılabilecektir.

Büyük veriye dağıtık dosyalama sistemi kullandığı için güvenliğini sağlamak güçtür. Saldırı gelebilecek birçok farklı yol vardır. Zeki teknikler kullanılarak geliştirilen makine öğrenmesi modeli ile yeni saldırılarıda keşfedilebilmek mümkün olabilmektedir. Örüntülerden yola çıkarak izinsiz girişler engellenebilmektedir.

Bu çalışmada büyük verileri işlemek için kullanılan büyük veri analitiği frameworklarından biri olan Apache Spark kullanılmıştır.

Apache Spark

Scala dili kullanılarak geliştirilmiş büyük veri üzerinde paralel işlem yapmayı sağlayan kütüphanedir. Daha hızlı olması sebebiyle Hadoop'tan farklılık göstermektedir. Hadoopta verinin yüklenmesi ve işlenmesine dair işlemler yapılırken, Spark'ta yüklenen veri üzerinde tekrar tekrar aynı işlemler yapılabilir. Spark üzerinde java, python, scala ve R ile akan veri ile uygulamalar yapılabilir. Spark SQL+Dataframes, MLlib, GraphX ve Spark Streaming gibi

kütüphaneleri vardır. Makine öğrenmesi, SQL, akan veri üzerinde çalışma olan “streaming” ve grafik işleme işlemleri bu modüller aracılığıyla gerçekleştirilir.

Apache Spark Ekosistemi

- **Spark Core:** Java, Scala ve Python API’leri kullanılarak çok çeşitli uygulamalar geliştirilir. Spark’ın üzerine inşa edildiği temel yürütme altyapısıdır.
- **Spark SQL + Dataframes:** Spark SQL modülü, yapısal veri işlemesinde kullanılır. Spark ekosisteminin geri kalanıyla güçlü bir entegrasyon sağlar.
- Streaming: Gerçek zamanlı olarak, akan yeni verileri işleme ve analiz etmeye yarar. Hem akan veriler hem de geçmiş veriler üzerinde çalışmayı sağlar.
- **MLlib(Machine Learning):** Makine öğrenmesi kütüphanesi olan MLLib, Java, Scala ve Python’da Spark uygulamalarının bir parçası olarak makine öğrenmesinde kullanılır.
- **Grafik Hesaplama (GraphX):** GraphX Spark’ın grafik hesaplama modülüdür.

RDD(resilient distributed dataset)

Esnek dağıtık veri kümeleridir. Spark veri yapısına veri almanın yollarından biridir.
Bir RDD, dağıtılmış bir öğe koleksiyonudur. Spark'taki tüm işler, yeni RDD'ler oluşturmak, mevcut RDD'leri dönüştürmek veya bir sonucu hesaplamak için RDD'lerde fonksiyon çağırmak olarak ifade edilir. Spark, RDD'lerde bulunan verileri kümeye otomatik olarak dağıtır ve bunlar üzerinde gerçekleştirilen işlemleri paralel hale getirir.

PYSPARK

PySpark, Apache Spark ve Python'un işbirliği ve birlikte kullanımı ile geliştirilmiş Spark için bir Python API'sidir. RDD'ler oluşturmaya ve MLlib kullanmaya yardımcı olur.

UYGULAMA GEREKLİLİKLERİ

1. Java jdk kurulu olmalı
2. Python kurulu olmalı
3. scala kurulumu yapılmalı
4. Apache spark indirilip kurulmalı
5. Gerekli yapılandırılmalar yapılmalı (.bashrc sonuna spark yolu, jupyter notebook fonksiyonu eklenmeli)
6. Pyspark kurulmalı (pip install pyspark)
7. Findspark kurulmalı (pip install findspark)
8. Anaconda Distrubution kurulu olup jupyter notebook çalışmalı.

UYGULAMA İÇİN PROBLEM ÇIKARIMI

Ağ izinsiz girişlerini tespit eden yazılım, bir bilgisayar ağını, belki içerdekiler de dahil olmak üzere yetkisiz kullanıcılardan korur. İzinsiz giriş algılayıcısını öğrenme görevi, izinsiz girişler veya saldırılar olarak adlandırılan `` kötü '' bağlantılar ile `` iyi '' normal bağlantılar arasında ayrım yapabilen öngörücü bir model (yani sınıflandırıcı) oluşturmaktır.

UYGULAMA İÇİN VERİSETİ

Bu çalışmada saldırı tespitinde sık kullanılan veriseti olan "KDD Cup 1999 Data" kullanılmıştır.

Bu veriseti 1998 DARPA Saldırı Tespiti Değerlendirme Programı, MIT Lincoln Labs tarafından hazırlanmıştır. Askeri bir ağ ortamında simüle edilen çok çeşitli izinsiz girişleri içeren, denetlenecek standart bir veri seti sağlanmıştır. 1999 KDD saldırı tespit yarışması bu veri kümesinin bir sürümünü Beşinci Uluslararası Bilgi Keşfi ve Veri Madenciliği Konferansı ile bağlantılı olarak düzenlenen Üçüncü Uluslararası Bilgi Keşfi ve Veri Madenciliği Araçları Yarışması için kullanılmıştır. Yarışma saldırı tespitinde yöntem ve uygulama geliştirmek için yapılmıştır. Yarışma, izinli ve izinsiz girişleri ayırabilen , saldırıları belirleyebilen öngörebilen bir ağ saldırı algılayıcısı oluşturmayı amaçlamaktadır.

Lincoln Laboratuvarları, ABD Hava Kuvvetleri yerel alan ağını simüle eden bir ortam oluşturmuş ve dokuz haftalık ham TCP döküm verilerini elde etmiştir. Simulasyona ayrıca birden fazla saldırıda eklemişlerdir. Bu dokuz haftalık verinin yedi haftalık ağ trafiğinden oluşan kısmı eğitim verisi, iki haftalık ağ trafiğinden oluşan kısmı test verileri şeklinde ayrılmıştır. Yaklaşık beş milyon eğitim verisi iken yaklaşık 2 milyon test verisidir.

Verisetinde bulunan dosyalar:

- **kddcup.names** Özniteliklerin listesidir
- **kddcup.data.gz** Tüm verisetidir.
- **kddcup.data_10_percent.gz** Full verisetinin yüzde 10 alt verisetidir.
- **kddcup.newtestdata_10_percent_unlabeled.gz** Etiketsiz yeni test verisetinin yüzde 10 alt verisetidir.
- **kddcup.testdata.unlabeled.gz** Etiketsiz test verisetidir.
- **kddcup.testdata.unlabeled_10_percent.gz** Etiketsiz test verisetinin yüzde 10 alt verisetidir.
- **corrected.gz** Doğrulanmış etiketle test verisetidir.
- **training_attack_types** Saldırı tip listesidir.
- **typo-correction.txt** Veri setinde düzeltilmiş bir yazım hatası hakkında nottur.

Öznitelikler:

1. **duration** Duration, kapsam [0, 58329]
2. **protocol_type** Protokol türü : TCP, UDP, ICMP
3. **service** Hedef ana bilgisayarin ag hizmeti turu, "http_443", "http_8001", "imap4" gibi 70 tür
4. **flag** Normal veya yanlış bağlantı durumu, ayrık tip, toplamda 11 tip, örneğin "S0", "S1", "S2" vb.
5. **src_bytes** Kaynak ana bilgisayardan hedef ana bilgisayara veri baytlarının sayisi, aralik [0,1379963888]
6. **dst_bytes** Hedef ana bilgisayardan kaynak ana bilgisayara veri baytlarının sayısı, aralik [0.1309937401]
7. **land** Ayni ana bilgisayardan / bağlanti noktasina teslim ediliyorsa 1, degilse 0
8. **wrong_fragment** Yanlis parça sayısı, sürekli tip, aralık [0,3]
9. **urgent** Acil paket sayısı, sürekli tip, aralık [0,14]
10. **hot** Sisteme duyarli dosyalara ve dizinlere erişim, aralik [0,101]
11. **num_failed_logins** Basarisiz giriş denemesi sayısı, aralık [0,5]
12. **logged_in** Basarili giriş için 1 degilse 0
13. **num_compromised** Guvenligi ihlal edilmis kosulun gerçeklesme sayisi, aralik [0,7479]
14. **root_shell** Root shell alinirsa 1, degilse 0
15. **su_attempted** su root komutu gorunurse 1 degilse 0
16. **num_root** root kullanicinin erisme sayisi aralik [0,7468]
17. **num_file_creations** Dosya olusturma islemlerinin sayisi, aralik [0,100]
18. **num_shells** shell komutunun kullanilma sayisi, aralik [0,5]
19. **num_access_files** Kontrol dosyasina erisim sayisi, aralık [0,9]
20. **num_outbound_cmd**s Bir FTP oturumundaki giden baglantilarin sayisi.Dataset' te görülme sayisi 0
21. **is_host_login** Giris host listesine mi aitse 1, yoksa 0
22. **is_guest_login** Misafir girisi ise 1,degilse 0
23. **count** Son iki saniyede, mevcut baglantiyla ayni hedef ana bilgisayara sahip baglantı sayisi, aralik [0,511]
24. **srv_count** Son iki saniyede, mevcut baglantiyla ayni hizmete sahip baglanti sayisi, aralık [0,511]
25. **serror_rate** Son iki saniye icinde, mevcut baglantiyla ayni hedef ana bilgisayara sahip baglantilar arasında, "SYN" hataları içeren

26. **srv_serror_rate** Son iki saniye içinde, mevcut baglantiyla ayni hizmete sahip baglantilar arasinda, "SYN" hatası olan bağlantıların yüzdesi, aralık [0.00,1.00]
27. **rerror_rate** Son iki saniye içinde, mevcut bağlantıyla aynı hedef ana bilgisayara sahip bağlantılar arasında, "RED" hatası içeren bağlantıların yüzdesi, aralık [0.00,1.00]
28. **srv_rerror_rate** Son iki saniye içinde, mevcut bağlantıyla aynı hizmete sahip bağlantılar arasında, "RED" hatası olan bağlantıların yüzdesi, aralık [0.00,1.00]
29. **same_srv_rate** Son iki saniye içinde, mevcut bağlantıyla aynı hedef ana bilgisayara sahip bağlantılar arasında, mevcut bağlantıyla aynı hizmete sahip bağlantıların yüzdesi, aralık [0.00,1.00]
30. **diff_srv_rate** Son iki saniye içinde, mevcut bağlantıyla aynı hedef ana bilgisayara sahip bağlantılar arasında, mevcut bağlantıya farklı hizmetler içeren bağlantıların yüzdesi, aralık [0.00,1.00]
31. **srv_diff_host_rate** Son iki saniye içinde, mevcut bağlantıyla aynı hizmete sahip bağlantılar arasında, mevcut bağlantıyla farklı bir hedef ana bilgisayara sahip bağlantıların yüzdesi, aralık [0.00,1.00]
32. **dst_host_count** İlk 100 bağlantıda, mevcut bağlantıyla aynı hedef ana bilgisayara sahip bağlantı sayısı, aralık [0,255]
33. **dst_host_srv_count** İlk 100 bağlantıda, aynı hedef ana bilgisayara ve mevcut bağlantıyla aynı hizmete sahip bağlantı sayısı, aralık [0,255]
34. **dst_host_same_srv_rate** İlk 100 bağlantıda, aynı hedef ana bilgisayara ve mevcut bağlantıyla aynı hizmete sahip bağlantıların yüzdesi, aralık [0.00,1.00]
35. **dst_host_diff_srv_rate** İlk 100 bağlantıda, mevcut bağlantıyla aynı hedef ana bilgisayara ve farklı hizmetlere sahip bağlantıların yüzdesi, aralık [0.00,1.00]
36. **dst_host_same_src_port_rate** İlk 100 bağlantıda, aynı hedef ana bilgisayara ve mevcut bağlantıyla aynı kaynak bağlantı noktasına sahip bağlantıların yüzdesi, aralık [0.00,1.00]
37. **dst_host_srv_diff_host_rate** İlk 100 bağlantıda, mevcut bağlantıyla aynı hedef ana bilgisayara ve aynı hizmete sahip bağlantıların yüzdesi, mevcut bağlantıdan farklı bir kaynak ana bilgisayara sahip bağlantıların yüzdesi, aralık [0.00,1.00]
38. **dst_host_serror_rate** İlk 100 bağlantıda, mevcut bağlantıyla aynı hedef ana bilgisayara sahip bağlantıların yüzdesi, SYN hataları olan bağlantıların yüzdesi, aralık [0.00,1.00]
39. **dst_host_srv_serror_rate** İlk 100 bağlantıda, mevcut bağlantıyla aynı hedef ana bilgisayara ve aynı hizmete sahip bağlantıların yüzdesi, SYN hataları olan bağlantıların yüzdesi, aralık [0.00,1.00]
40. **dst_host_rerror_rate** İlk 100 bağlantı arasında, mevcut bağlantıyla aynı hedef ana bilgisayara sahip bağlantıların yüzdesi, RED hatası içeren bağlantıların yüzdesi, aralık [0.00,1.00]

41. **dst_host_srv_rerror_rate** İlk 100 bağlantıda, aynı hedef ana bilgisayara ve mevcut bağlantıyla aynı hizmete sahip bağlantıların yüzdesi REJ hataları içeriyor, aralık [0.00,1.00]
42. **label** Saldırı etiketleri

Etiketler:

1. **normal**
2. **back** dos
3. **buffer_overflow** u2r
4. **ftp_write** r2l
5. **guess_passwd** r2l
6. **imap** r2l
7. **ipsweep** probe
8. **land** dos
9. **loadmodule** u2r
10. **multihop** r2l
11. **neptune** dos
12. **nmap** probe
13. **perl** u2r
14. **phf** r2l
15. **pod** dos
16. **portsweep** probe
17. **rootkit** u2r
18. **satan** probe
19. **smurf** dos
20. **spy** r2l
21. **teardrop** dos
22. **warezclient** r2l
23. **warezmaster** r2l

Sonuç olarak verseti 41 öznitelik ve sınıf etiketinden oluşur. Sınıf etiketleri ise 23 çeşittir ancak uygulamada normal olan harici saldırı olarak işaretlenmiştir. Saldırı türü değil saldırı olup olmaması tespit edilmiştir.

UYGULAMA

1)SPARK TANITMA

İşletim sistemi linux olduğu için os kütüphanesi eklenir

import os

Sparkı bulma kütüphanesi eklenir

import findspark

Jupyter notebooka sparkı kurduğumuz adres verilir sparkı bulması için.

findspark.init("/usr/local/spark/spark-3.0.1-bin-hadoop2.7/")

Pysapark ile sparka bağlanılır ve spark session başlatılır.

from pyspark import SparkContext

from pyspark.sql import SparkSession

2)VERİ ÇEKME

Veri çekmek için urllib kütüphanesi eklenir.

import urllib

UCI Makine öğrenmesi veri kaynağından KDD Cup 1999 eğitim veriseti dosyası "kddcup.data.gz" çekilir

f = urllib.request.urlretrieve("http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data.gz", "kddcup.data.gz")

Veri değişkene atanır.

data_file = "./kddcup.data.gz"

Eğitim veriseti kontrol edilir.

data_file

Eğitim verisi için RDD veriyapısı oluşturulur. Spark'ın textFile fonksiyonu sıkıştırılmış dosyaları doğrudan işlemeyi sağlamaktadır.

raw_data = sc.textFile(data_file)

#eğitim verisi RDD oluşmuşmu kontrol edilir.

raw_data

```
In [22]: #eğitim verisi RDD
         raw_data

Out[22]: ./kddcup.data.gz MapPartitionsRDD[3] at textFile at NativeMethodAccessorImpl.java:0
```

RDD ye yüklenen satır sayısını saydırılır.

raw_data.count()

4898431 yani yaklaşık 5 milyon satır eğitim verisi vardır.

Eğitim verisinden ilk 10 satır görüntülenir.

raw_data.take(10)

```
Out[35]: ['0,tcp,http,SF,215,45076,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0,0,0.00,0.00,0.0
0,0.00,0.00,0.00,0.00,0.00,normal.',
 '0,tcp,http,SF,162,4528,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,1,1.00,0.00,1.0
0,0.00,0.00,0.00,0.00,0.00,normal.',
 '0,tcp,http,SF,236,1228,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,2,2,1.00,0.00,0.5
0,0.00,0.00,0.00,0.00,0.00,normal.',
 '0,tcp,http,SF,233,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,3,3,1.00,0.00,0.3
3,0.00,0.00,0.00,0.00,0.00,normal.',
 '0,tcp,http,SF,239,486,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,3,0.00,0.00,0.00,0.00,1.00,0.00,0.00,4,4,1.00,0.00,0.25,
0.00,0.00,0.00,0.00,0.00,normal.',
 '0,tcp,http,SF,238,1282,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,4,0.00,0.00,0.00,0.00,1.00,0.00,0.00,5,5,1.00,0.00,0.2
0,0.00,0.00,0.00,0.00,0.00,normal.',
 '0,tcp,http,SF,235,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.00,0.00,0.00,0.00,1.00,0.00,0.00,6,6,1.00,0.00,0.1
7,0.00,0.00,0.00,0.00,0.00,normal.',
 '0,tcp,http,SF,234,1364,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,7,7,1.00,0.00,0.1
4,0.00,0.00,0.00,0.00,0.00,normal.',
 '0,tcp,http,SF,239,1295,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,7,7,0.00,0.00,0.00,0.00,1.00,0.00,0.00,8,8,1.00,0.00,0.1
2,0.00,0.00,0.00,0.00,0.00,normal.',
 '0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.1
1,0.00,0.00,0.00,0.00,0.00,normal.']
```

UCI Makine öğrenmesi veri kaynağından KDD Cup 1999 test veriseti dosyası "corrected.gz" çekilir

ft = urllib.request.urlretrieve("http://kdd.ics.uci.edu/databases/kddcup99/corrected.gz", "corrected.gz")

Test veriseti değişkene atanır.

test_data_file = "./corrected.gz"

Test verisinden RDD veriyapısı oluşturulur

test_raw_data = sc.textFile(test_data_file)

Test veri dosyası RDD kontrol edilir

test_raw_data

```
In [29]: #test veri dosyası RDD
         test_raw_data

Out[29]: ./corrected.gz MapPartitionsRDD[10] at textFile at NativeMethodAccessorImpl.java:0
```

Test verisi satır sayısı saydırılır.

test_raw_data.count()

311029 satır test verisi vardır.

Test verisinden ilk 10 satır görüntülenir.

test_raw_data.take(10)

```
Out[34]: ['0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.0
         1,0.00,0.00,0.00,0.00,0.00,0.00,normal.',
          '0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.0
         1,0.00,0.00,0.00,0.00,0.00,0.00,normal.',
          '0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.0
         1,0.00,0.00,0.00,0.00,0.00,0.00,normal.',
          '0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.0
         1,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.',
          '0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.0
         1,0.01,0.00,0.00,0.00,0.00,0.00,snmpgetattack.',
          '0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.0
         0,0.01,0.00,0.00,0.00,0.00,0.00,snmpgetattack.',
          '0,udp,domain_u,SF,29,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,0.00,0.00,0.00,0.00,0.50,1.00,0.00,10,3,0.30,0.30,0.3
         0,0.00,0.00,0.00,0.00,0.00,normal.',
          '0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.0
         1,0.00,0.00,0.00,0.00,0.00,0.00,normal.',
          '0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.0
         1,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.',
          '0,tcp,http,SF,223,185,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,4,0.00,0.00,0.00,0.00,1.00,0.00,0.00,71,255,1.00,0.00,0.
         01,0.01,0.00,0.00,0.00,0.00,normal.']
```

3)VERİ ÖNİŞLEME

MLlib makine öğrenmesi kütüphanesi ve etiketleme kütüphanesi eklenir.

from pyspark.mllib.regression import LabeledPoint

Dizi işlemleri için numpy eklenir.

from numpy import array

Satırdaki etiketi normalse saldırı değil, farklı ise saldırı olarak etiketleyen, öznitelik seçen fonksiyon yazılır. normal:0 attack: 1 şeklinde sayısallaştırma yapılır. İlk 4 öznitelik veriye dahil edilmez. Yani duration, protocol_type, service, flag veriden çıkartılır.

def parse_interaction(line):

line_split = line.split(",")

```
    clean_line_split = line_split[0:1]+line_split[4:41]
    attack = 1.0
    if line_split[41]=='normal.':
        attack = 0.0
    return LabeledPoint(attack, array([float(x) for x in clean_line_split]))
```

Eğitim verisini fonksiyona görehazırlama

```
training_data = raw_data.map(parse_interaction)
```

Test verisini fonksiyona göre hazırlama

```
test_data = test_raw_data.map(parse_interaction)
```

4)VERİ ANALİZİ

Binary logistic regression classification kütüphanesi eklenir. Etiketler 0 ve 1 şeklinde binary yapılmıştı.

```
from pyspark.mllib.classification import LogisticRegressionWithLBFGS
```

Eğitim verisinin lojistik regresyonla eğitilmesi gerçekleştirilir. Model oluşturulur.

```
logistic_model = LogisticRegressionWithLBFGS.train(training_data)
```

!Büyük veri kullanıldığı için eğitim işlemi 2 saatten fazla sürmüştür.

5)MODELİ TEST VERİSİNE UYGULAMA

Model ile test verisine tahmin yapılır.

```
prediction = test_data.map(lambda p: (p.label, logistic_model.predict(p.features)))
```

Tahmin veriyapısı kontrol edilir.

```
prediction
```

```
In [112]: prediction
Out[112]: PythonRDD[383] at RDD at PythonRDD.scala:53
```

6)MODEL BAŞARISI HESAPLAMA

Model doğruluğu hesaplamak için sklearn metrics ve numpy kütüphaneleri eklenir. (Burada MLlib in kendi metrikleride kullanılabilirdi ama çalıştıramadım bende sklearn metriği ekledim.)

```
from sklearn.metrics import accuracy_score
```

```
import numpy as np
```

Accuracy (doğruluk) hesaplanır

```
predictionAndTargetNumpy = np.array((prediction.collect()))
```

```
acc = accuracy_score(predictionAndTargetNumpy[:,0], predictionAndTargetNumpy[:,1])
```

Accuracy i kontrol edilir

```
acc
```

```
In [128]: acc
Out[128]: 0.8625819457349636
```

SONUÇ

Bu çalışmada büyük veride saldırı tespit sistemi uygulaması gerçekleştrilmiştir. Apache spark kurularak dağıtık dosya sisteminde KDD 99 veriseti RDD veri yapısına çevrilerek paralel işlemler gerçekleştirilmiştir. Pyspark ve MLlib kütüphaneleri ile binary lojistik regresyon algoritması ile veriler eğitilerek model geliştirilmiş, test verisi ile tahmn yapılmıştır. Sonuç olarak %86 doğruluk oranı elde edilmiştir. Bu doğruluk oranı ile modelin büyük veride saldırı tespitinde başarılı olduğu sonucu çıkarılır.

KAYNAKLAR

1. APACHE SPARK https://spark.apache.org/
2. Pyspark https://spark.apache.org/docs/latest/api/python/index.html
3. MLlib https://spark.apache.org/docs/latest/mllib-data-types.html#labeled-point
4. Ubuntu Apache spark kurulum https://intellitech.pro/spark-installation-on-ubuntu/
5. Ubuntu Apache spark kurulum https://www.udemy.com/tutorial/scala-and-spark-for-big-data-and-machine-learning/linux-ubuntu-setup-and-installation/?utm_source=adwords&utm_medium=udemyads&utm_c

ampaign=DSA_Catchall_la.EN_cc.ROW&utm_content=deal4584&utm_term=_._ag_88010211481_._ad_437497337004_._kw__._de_c_._dm__._pl__._ti_dsa-39880105563_._li_9056891_._pd__._&matchtype=b&gclid=CjwKCAiAi_D_BRApEiwASslbJ7zWRKN20L5aogthJ147OIwVo-1zt78PDCWtm_-hbEXC3B7qimP01xoCcTIQAvD_BwE

6. Veriseti http://kdd.ics.uci.edu/databases/kddcup99/kddcup99