Ad-Soyad: Meryem Idrıs

No:170504068

Ad-Soyad:Mehemt Fevzi Canbek

No:190504103

Tanım

Veri kümesinin amacı, normal trafik ve arka plan trafiğiyle karışık gerçek botnet trafiğinin büyük bir kısmını yakalamaktı.

CTU-13 veri seti, farklı botnet örneklerinin on üç yakalamasından (senaryo adı verilen) oluşur. Her senaryoda, çeşitli protokoller kullanan ve farklı eylemler gerçekleştiren belirli bir kötü amaçlı yazılım çalıştırdık.

Uzun Açıklama

Her senaryo, üç trafik türünün tüm paketlerini içeren bir pcap dosyasında yakalandı. Bu pcap dosyaları, NetFlows, WebLogs vb. gibi diğer türdeki bilgileri elde etmek için işlendi.

CTU-13 veri kümesinin "An ampirik karşılaştırması botnet algılama yöntemleri" makalesinde açıklanan ve yayınlanan ilk analizinde, trafiği temsil etmek ve etiketleri atamak için tek yönlü NetFlow'lar kullanıldı.

Bu tek yönlü NetFlow'lar kullanılmamalıdır çünkü çift yönlü NetFlow'ların kullanıldığı veri kümesine yönelik ikinci analizimizde daha iyi performans gösterdiler. Çift yönlü NetFlow'ların yönlü olanlara göre birçok avantajı vardır.

Birincisi, istemci ve sunucu arasındaki ayrım sorununu çözüyorlar, ikincisi daha fazla bilgi içeriyorlar ve üçüncüsü çok daha ayrıntılı etiketler içeriyorlar. Veri setinin çift yönlü NetFlow'larla ikinci analizi burada yayınlanmıştır.

Senaryonun süresi, paket sayısı, NetFlow sayısı ve pcap dosyasının boyutu arasındaki ilişki Tabloda gösterilmektedir. Bu Tablo aynı zamanda yakalamayı oluşturmak için kullanılan kötü amaçlı yazılımı ve virüslü bilgisayarların sayısını da gösterir. her senaryo.

-Tek Yönlü ve Çift Yönlü NetFlow Nedir?

A ana bilgisayarından B ana bilgisayarına trafik akışı, iki yönde paket alışverişinden oluşur (A->B ve B->A<cevap>). Bunlar iki farklı tek yönlü akış olarak kabul edilir.

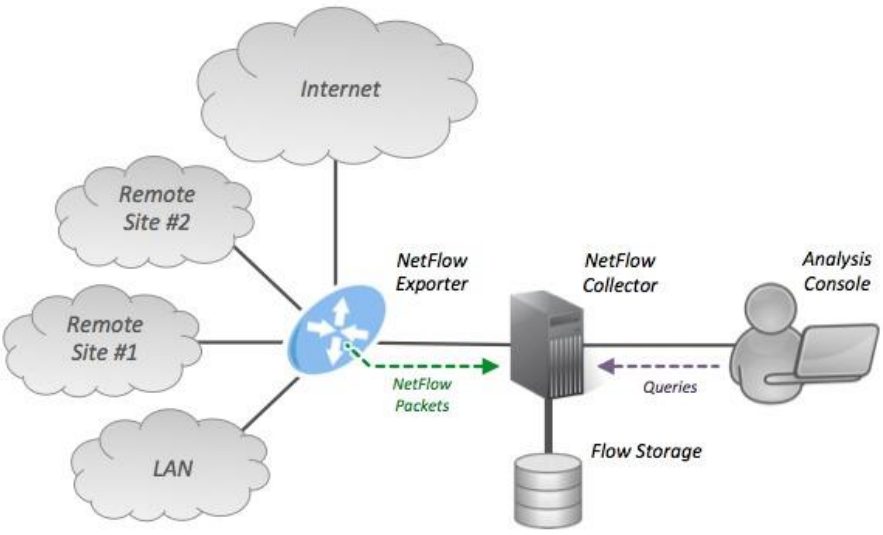
Palo Alto Networks güvenlik duvarları, PA-4000 Serisi hariç, tek yönlü NetFlow'u destekler. Güvenlik duvarı, A'dan B'ye ve B'den A'ya trafik akışını görmeyi bekler.

Bununla birlikte, NAT söz konusu olduğunda, akış A'dan B'ye ve B'den A'nın doğal IP'sine doğru görünür.

Çift Yönlü NetFlow:

A ana bilgisayarından B ana bilgisayarına trafik akışı, A->B'den yalnızca bir akış olarak kabul edilir. Bu, A->B ve B->A'dan gelen paketlerin toplamını içerir.

Çift Yönlü NetFlow, Palo Alto Networks cihazlarında desteklenmez.



```
#!/usr/bin/python
import os
import sys
import glob
from datetime import timedelta, date

#
##
###
#### By Evan Daman
####
#### CAO: 20191106
###
##
#

try:
    uFile = sys.argv[1]
except:
```

```
    print("\nconvert-to-bidirectional-flow.py combines uni-directional SILK flows into bi-directional flows for easier analysis")
    print("Usage: python convert-to-bidirectional-flow.py inputfile.txt\n")
    print("\nPlease specify a text file containing ascii silk flow records")
    print("Your pwd must be the location of the input file")
    print("\nIMPORTANT: For the most accurate results follow these commands to make your txt file.\n")
    print("rwsort --fields=9 <flowrecord.bin> | rwcut >> <silkflows.txt>")
    print("uniq <silkflows.txt> >> <filename.txt>")
    print("\nTIP: prefix your command with LC_ALL=C to improve speed")
    quit()

if (uFile == "-h" or uFile == "--help"):
    print("\nconvert-to-bidirectional-flows.py combines uni-directional SILK flows into bi-directional flows for easier analysis")
    print("Usage: python convert-to-bidirectional-flow.py inputfile.txt\n")
    print("\nPlease specify a text file containing ascii silk flow records")
    print("Your pwd must be the location of the input file")
    print("\nIMPORTANT: For the most accurate results follow these commands to make your txt file.\n")
    print("rwsort --fields=9 <flowrecord.bin> | rwcut >> <silkflows.txt>")
    print("uniq <silkflows.txt> >> <filename.txt>")
    print("\nTIP: prefix your command with LC_ALL=C to improve speed")
    quit()

if not os.path.isfile(sys.argv[1]):
    print(uFile + " does not exist.")
```

```
    print("\nconvert-to-bidirectional-flows.py combines uni-directional SILK flows into bi-directional flows for easier analysis")
    print("Usage: python convert-to-bidirectional-flow.py inputfile.txt\n")
    print("\nPlease specify a text file containing ascii silk flow records")
    print("Your pwd must be the location of the input file")
    print("\nIMPORTANT: For the most accurate results follow these commands to make your txt file.\n")
    print("rwsort --fields=9 <flowrecord.bin> | rwcut >> <silkflows.txt>")
    print("uniq <silkflows.txt> >> <filename.txt>")
    print("\nTIP: prefix your command with LC_ALL=C to improve speed")
    quit()

print("\nIf the script encounters an error. Please read the help statement\n")


#Function to iterate through date for use in filename
def daterange(start_date, end_date):
    for n in range(int ((end_date - start_date).days)):
        yield start_date + timedelta(n)



#Open user supplied text flow
uFlow = open(uFile)
uLine = uFlow.readlines()

dateArray = []
```

```
for line in uLine:
    line = line.replace(" ","")
    line = line.replace("|",",")
    line = line.rstrip()
    dateTemp = line.split(",")
    dateVal = dateTemp[10].split("T")
    if dateVal[0] not in dateArray:
        dateArray.append(dateVal[0])
dateArray.sort(reverse=True)


date2 = dateArray[1]
date1 = dateArray[-1]
print("Start date: " + str(date1))
print("End date: " + str(date2))


date1 = date1.split("/")
date2 = date2.split("/")


#Init time scope
start_date = date(int(date1[0]), int(date1[1]), int(date1[2]))
end_date = date(int(date2[0]), int(date2[1]), int(date2[2]) + 2)


for single_date in daterange(start_date, end_date):
```

```
    fileList = glob.glob(single_date.strftime("%Y-%m-%d") + "-" + uFile + "-*")
    for filepath in fileList:
        if os.path.isfile(filepath):
            os.remove(filepath)
fileList = []


fNames = []


#Break out Input file by day
for single_date in daterange(start_date, end_date):
    #Print date to show progress
    print(single_date.strftime("%Y-%m-%d"))
    f = 0
    i = 1
    for line in uLine:
        line = line.replace(" ", "")
        line = line.replace("|", ",")
        line = line.rstrip()
        lineArrayDay = line.split(",")


        #If our date is in the line
        if single_date.strftime("%Y/%m/%d") in lineArrayDay[10]:
            if i == 10000:
                f = f + 1
```

```
            i = 1
        if i < 10000:
            fileName = single_date.strftime("%Y-%m-%d") + "-" + uFile + "-" +
str(f)
            open(fileName, "a").write(line + "\n")
            i = i + 1
            if fileName not in fNames:
                fNames.append(fileName)


#Open the final result file
results = open("Bidirectional-" + uFile, "a")
results.write("src" + "," + "sport" + "," + "dest" + "," + "dport" + "," + "bytes S-
>D" + "," + "bytes S<-D" + "," + "bytesTotal" + "," + "packets S->D" + "," +
"packets S<-D" + "," + "packetTotal" + "," + "srcFlags" + "," + "destFlags" + "," +
"startTime" + "," + "endTime" + "\n")



#Create Bi-Directional flow records
for fName in fNames:
    #Display script progress
    print(fName)
    if os.path.isfile(fName):
        dayFile = open(fName, "a+")
        dayLine = dayFile.readlines()

        #Initialize our list that will hold previous positive hits
```

```
    skipList = []

    #Init a skip var so we can keep going through the file until all lines have
been added to skipList
    b = 0
    previousPos = -1
    i = 0
    r = 1

    while (i < len(dayLine) and b == 0):
       if (r == 1):
          src = "src"
          sport = 0
          dest = "dest"
          dport = 0
          bytesAB = 0
          bytesBA = 0
          bytesTotal = 0
          packetAB = 0
          packetBA = 0
          sFlags = "null"
          dFlags = "null"
          packetTotal = 0
          endTime = "null"
          startTime = "null"
          r = 0
          s = 0
```

```
        b = 0
        o = 0


#0-src
#1-dest
#2-sport
#3-dport
#5-packets
#6-bytes
#7-flags
#8-startTime
#10-endTime


    if dayLine[i] not in skipList:
        lineArray = dayLine[i].split(",")
        if (s == 0):
            s = 1
            src = lineArray[0]
            sport = int(lineArray[2])
            startTime = lineArray[8]
            sFlags = lineArray[7]
            endTime = lineArray[10]
            bytesAB = bytesAB + int(lineArray[6])
            bytesTotal = bytesTotal + int(lineArray[6])
            packetAB = packetAB + int(lineArray[5])
```

```
            packetTotal = packetTotal + int(lineArray[5])
            dest = lineArray[1]
            dport = lineArray[3]
            otherSide = dest + "," + src + "," + str(dport) + "," + str(sport)
            previousPos = i

        if (otherSide in dayLine[i]):
            skipList.append(dayLine[i])
            packetBA = packetBA + int(lineArray[5])
            packetTotal = packetTotal + int(lineArray[5])
            bytesBA = bytesBA + int(lineArray[6])
            bytesTotal = bytesTotal + int(lineArray[6])
            endTime = lineArray[10]
            dFlags = lineArray[7]
            o = 1
            r = 1

        #If we've seen a src/dest write immediatly and reset
        #If we've seen only a src this whole file(no dest flow). write and reset.
        if (s == 1 and ((o == 1) or (dayLine[i] == dayLine[-1]))):
            results.write(str(src) + "," + str(sport) + "," + str(dest) + "," + str(dport)
+ "," + str(bytesAB) + "," + str(bytesBA) + "," + str(bytesTotal) + "," +
str(packetAB) + "," + str(packetBA) + "," + str(packetTotal) + "," + str(sFlags) +
"," + str(dFlags) + "," + str(startTime) + "," + str(endTime) + "\n")
            r = 1
            i = previousPos
```

```
        #Break out of this day's file if no new lines are encountered and a line
hasnt been written this loop
        if ((s == 0) and (r == 0) and dayLine[i] == dayLine[-1]):
            b = 1


        i = i + 1




dayFile.close()
for fName in fNames:
    os.remove(fName)
if os.path.isfile("Bidirectional-" + uFile):
    print("File Bidirectional-" + uFile + " created!")
```


Soyut

Siber uzay altyapısının hayati bir unsuru siber güvenliktir. Siber uzayın ilgili altyapısını etkileyen anormalliklere yol açan güvenlik sorunları için birçok protokol önerilmiştir. Mobil cihazlardaki anormal davranışları azaltmak için kullanılan makine öğrenimi (ML) yöntemleri. Bu makale, iki kötü amaçlı yazılım veri kümesindeki olası anormallikleri tespit etmek için Yüksek Performanslı Aşırı Öğrenme Makinesi'ni (HP-ELM) uygulamayı amaçlamaktadır.

HP-ELM'nin etkinliğini test etmek için yaygın olarak kullanılan iki veri kümesi (CTU-13 ve Kötü Amaçlı Yazılım) kullanılır. HP-ELM öğrenme yönteminin etkinliğini doğrulamak için kapsamlı karşılaştırmalar yapılır. Deney sonuçları, HP-ELM'nin, bir aktivasyon fonksiyonuyla ilk 3 özellik için 0,9592 ile en yüksek performans doğruluğuna sahip olduğunu göstermektedir.

Anahtar Kelimeler:

Öğrenme, Kötü Amaçlı Yazılım Tespiti, Yüksek Performanslı Ekstrem Öğrenme Makinesi (HP-ELM), Mobil Uygulamalar, Nesnelerin İnterneti (IoT).

1. Giriş

Akıllı telefonların popülaritesi son zamanlarda artıyor. Gartner'a göre, akıllı telefon kullanıcılarının toplam sayısı 2020 yılına kadar altı milyara ulaşacak (Ericsson'a göre [1]), ayrıca kötü amaçlı mobil kötü amaçlı yazılımlar tarafından 40 milyondan fazla saldırı (Kaspersky Labs'e göre [2]) ve veri ihlali olacak 2020 yılına kadar 150 milyon doları aşacak (Juniper araştırmasına göre [3]). Akıllı telefon cihazları, modüllerini sistem dizinine yükleyen ve tipik sistem davranışını bozmaya çalışan kötü amaçlı kötü amaçlı yazılımlardan etkilenir [4].

Bu sorunlarla başa çıkabilmek için araştırmacılar ve güvenlik analistleri, mobil cihazlara yönelik güvenilir uygulamalar oluşturmak amacıyla pratik veya bilimsel amaçlı bir çalışma yürüttüler. [5]'te Russon, 3,2 milyondan fazla indirilen 104'ten fazla Google Play uygulamasında çeşitli gizli kötü amaçlı yazılım türleri keşfetti. Kullanıcının mobil cihazlarına yapılan çok sayıda saldırının sistemdeki CPU yüklerini etkilemesine neden olur[5].

Kötü amaçlı yazılım tespiti için çeşitli güvenlik protokolleri uygulandı. Bu tür sistemler anomali bazlı veya davranış bazlı olabilir. İlki önceden tanımlanmış bir modele dayanır. Bu tür İzinsiz Giriş Tespit Sistemi (IDS), ağ donanımının taramalarını ve araştırmalarını tanımlamak için etkili bir yaklaşımdır ve Telnet gibi saldırıları başlatmadan önce potansiyel izinsiz girişlere ilişkin erken uyarı ipuçları verir. Bu tür sistemler düzenli imza güncellemelerinin (yani imza veritabanının kapsamının) alınmasına bağlıdır. Mirai bot ağı ve türevleri tarafından yapılan DDoS saldırılarının dramatik etkisi, IoT cihazları için riskleri vurgulamaktadır [6]. Datagram Aktarım Katmanı Güvenliği (DTLS) protokolü adı verilen uçtan uca güvenlik şeması, bir şifreleme tekniği kullanır [7]. DTLS, belirli saldırı türlerini azaltsa da, her tür büyük veriye dayalı anormal davranışı tanımlamada başarısız olur. Önemli dezavantajı ise saldırıya özel olmalarıdır [8]. Bu nedenle, IoT ortamlarında güvenlik ve gizlilik konusunun verimliliğini artırmak için IoT cihazlarının belirsizliğini ve akıllı karar verme potansiyel kapasitesini ölçecek metodolojiler ve prosedürler geliştirmek esastır. Eğik

algoritmalar kullanarak dolandırıcılık veya güvenlik açığı tespiti açısından akıllı öneriler sunmak da önemlidir.

IoT tabanlı kötü amaçlı yazılım tespitindeki güvenlik sorununu azaltmak için nöro bulanıklık gibi yumuşak hesaplama teknikleri önerilmiştir [9]. Neuro-Fuzzy sınıflandırıcıları, statik verilerden ve uygulamaların yürütme sırasında oluşturduğu verilerden eğitilir. Neuro-Fuzzy, toplanan istatistiklere ve türetilmiş bulanık kurallara dayalı olarak kötü amaçlı yazılım tespitinde potansiyele sahiptir ([10], [11], [12]). Ayrıca, kötü amaçlı yazılım kodlarını ve davranışlarını tanımlamak ve belirli tehdit türlerini tanımlamak için çeşitli öğrenme algoritmaları önerilmiştir ([13],[14]). Nöro bulanıklık gibi yumuşak hesaplama tekniklerinin temel dezavantajı, bulanık kuralların sinir ağının ağırlıklarını ayarlamak için rastgele üretilmesi ve uygulamanın türüne bağlı olarak gizli katman sayısının artabilmesidir. Ancak ayarlama prosedürünü hızlandırmanın bir yolu yoktur. Böylece ELM, Tek Gizli Katmanlı İleri Beslemeli Sinir Ağlarında (SLFN'ler) ([15], [16], [17]) ve [18] aktivasyon fonksiyonlarını ayarlayabilir.

Bu yazıda, eğitim verilerinden bağımsız olarak rastgele oluşturulan gizli nöronların parametrelerini çağırmak için yöntemin öğrenme aşamasına yardımcı olmak amacıyla Yüksek Performanslı Ekstrem Öğrenme Makinesi (HP-ELM) adı verilen hızlı yakınsak bir pekiştirme çözümünden faydalandık. 19]. Ayrıca böyle bir yöntem, çeşitli ölçeklerdeki veri setlerini, farklı yapı seçim seçeneklerini ve düzenlileştirme yöntemlerini test etmek için kullanılır.

Çalışmamızda kötü amaçlı yazılımları iki veri kümesinde sınıflandırmak için HP-ELM uygulanmıştır.

Çalışmamızda şu araştırma sorularını ele alıyoruz: 1. "IoT tabanlı kötü amaçlı yazılım cihazlarının anlamlı değerlerini elde etmek için kullanılan mevcut veri analizi teknikleri nelerdir?", 2. "IoT üzerinde kötü amaçlı yazılımların etkisi nasıl korunur?" ve 3. “Önerilen güvenlik ve mahremiyet koruma çerçevesinin büyük veri platformunda ölçeklenebilirlik açısından etkisi nedir?”. Çalışmamız adli kötü amaçlı yazılım davranışına katkıda bulunmaktadır.

Bu çalışmada kullanılan mobil kötü amaçlı yazılım uygulamalarının önceki sonuçları ve veri kümeleri ([20] ,[21]).

Makalenin katkısı, IoT kötü amaçlı yazılımlarını tanımak için aşağıdaki gibi bir IDS yöntemi önermektir.

• İki kötü amaçlı yazılım veri kümesinde bir özellik seçme yöntemi uyguladık.

• IoT ortamı için HP-ELM sınıflandırıcıyı temel alan bir kötü amaçlı yazılım tespit sistemi geliştirdik.

• Önerilen sistemin verimliliğini iki kıyaslama veri kümesini kullanarak test ettik: CTU-13 veri kümesi [20] ve DyHAP kötü amaçlı yazılım veri kümesi [21] ve HP-ELM'nin performansını özellik seçimiyle ve özellik seçimi olmadan karşılaştırdık.

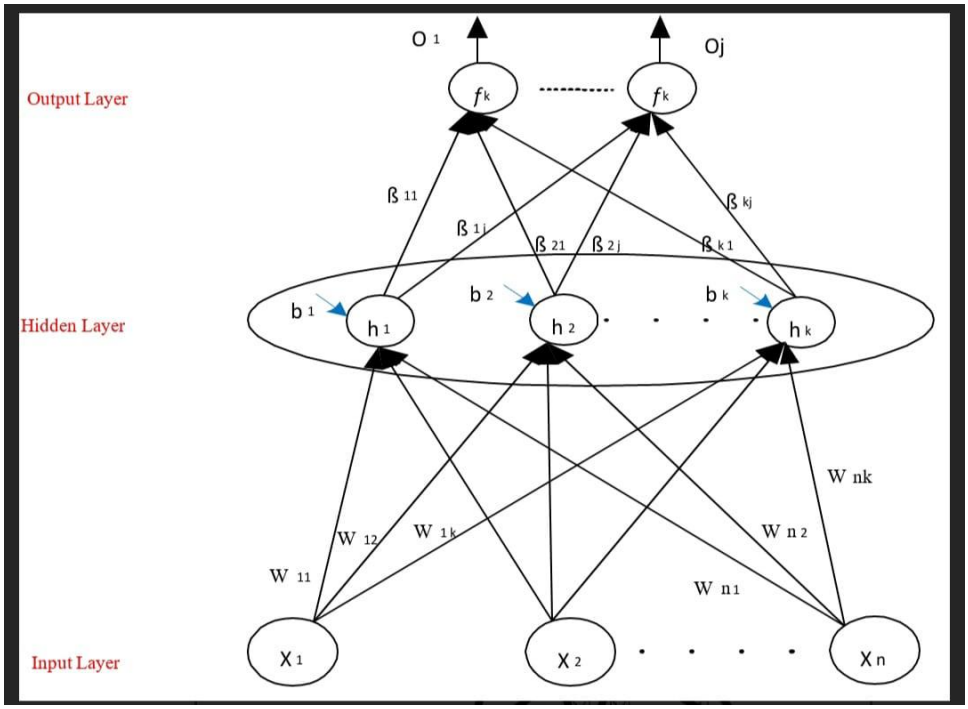
2. Önceki İş Bu bölüm, kötü amaçlı yazılım temelli iot için güvenlik çerçevesini kullanan teknik makaleler sunar [22,23]. [22] 'de, bir uygulama programı, gerçek uygulamanın (yani, ters bir mühendis işleminin) uygulanmasıyla, [23]' de uygulanmasıyla, bir program, bir program uygulamayı yürüterek çalışan işlemlerin davranışını araştırır. Bir yandan, statik tip bir kötü amaçlı yazılım programı düşük bellek kaynakları gerektirir, minimum CPU işlemleri ve analiz işlemi hızlıdır, diğer taraftan, bilinmeyen değişiklikleri ve kötü amaçlı yazılımın varlığını tespit etmek için dinamik bir tane kullanılabilir [24]. Özel ve bir kamu veri setini kullanan [25] 'de bir makine öğrenme tabanlı kötü amaçlı yazılım algılama önerdi. Son olarak, çözümlerini çeşitli durumlarda doğruladılar. Kağıdımızda, önerilen yöntemleri özel bir (Andoird kötü amaçlı yazılım) ve farklı bir HP-ELM parametresi ayar deneylerini kullanarak bir kamu (CTU-13) veri kümesinde de değerlendirdik. [24] 'deki yazarlar, uygulamanın davranışlarının akışlarına ulaşmak için bir crowdsourcing sistemi kullanır. Tek bir işletim sistemine dayanarak, tek bir işletim sistemine dayanan cep telefonu kötü amaçlı yazılımlarda birçok çalışma yapılmıştır. [22] 'deki yazarlar, 125 mevcut kötü amaçlı yazılımlı aileyi kullanarak çok düzeyli ve davranış tabanlı bir Android kötü amaçlı yazılım tespiti sunar ve% 96'yı kötü amaçlı yazılımın algılanmasını bildirir. Bununla birlikte, bu yaklaşım daha az anlamsal bilgi içeren ve kötü niyetli davranışları doğru bir şekilde tespit edemeyen sistem çağrıları uygular. Son zamanlarda, [26] 'ndaki yazarlar, çalışma zamanı davranışını çalıştıran sistem çağrıları ve "bağlayıcı işlemler". Bu yöntem, mantık yapılarını ve mobil uygulamaların çalışma zamanı davranışlarını android cihazlar için ortaklaşa kapsayan yeni bir kötü amaçlı yazılımlı değişken algılama istemci-sunucu sistemi sunar. Ancak, bu yaklaşım ağ durumuna bağlı, grafik madenciliği; Hesaplama karmaşıklığını doğrudan etkileyen ağ performansını etkiler. Bu nedenle, gerçek zamanlı algılama için uygun değildir. [27]

Yazarlarda, mobil cihazlardan kötü niyetli davranışları tanımlamak için hafif bir algılama sistemi sunar. İstatistiksel Markov Zincir Modelleri Uygulama davranışı bir özellik vektörü biçiminde oluşturmak için uygulanan ve rastgele orman, uygulama davranışını sınıflandırmak için kabul edilir. Sonuçlar% 96'lık bir doğruluk gösterir. Mirai, yüksek profilli hedefleri etkileyen olağanüstü DDOS saldırılarını sınıflandırmaya çalışır [6]. IOT cihazlarındaki kontrol için bir uyandırma çağrısıdır ve giderek daha fazla DDOS saldırıları riskini analiz eder. Olası bir kullanıcı adı ve şifre çiftlerinin küçük bir sözlüğüne dayanarak kaba kuvvet kullanarak iot cihazlarının idari kimlik bilgilerini azaltır [28]. Bu tür sorunlarla başa çıkmak için, IOT ortamında mevcut olmayan kaynakları göz önüne alarak, DDOS'a karşı verimli bir şekilde koruma sağlayabilecek yeni güvenlik çözümleri olarak kaynak verimliliği hibrit kimliklerine ihtiyacımız var.

3. HP-ELM tespit yöntemi

Yumuşak hesaplama tekniğinin en büyük dezavantajı, sinir ağının ağırlıklarını ayarlamak için bulanık kuralların rastgele oluşturulması ve uygulamanın türüne bağlı olarak gizli katman sayısının artabilmesidir. Ancak ayarlama prosedürünü hızlandırmanın bir yolu yoktur. Böylece ELM, tek gizli katmanlı ileri beslemeli sinir ağlarında (SLFN'ler) ([15], [16], [17]) ve [18] aktivasyon fonksiyonlarını ayarlayabilir.

Bu bölümde, iki kötü amaçlı yazılım veri kümesi Android Malware [21] ve CTU-13 veri kümesine [20] uygulanan HP-ELM mobil ağ mimarisi sunulmaktadır. Şekil 1'de belirtildiği gibi ELM, SLFN ağları için hızlı eğitim yöntemidir.


O 1
Oj
Output Layer
fk
fk
ß 11
ß 1j
ß 21
ß 2j
ß kj
ß k1
b 1
b 2
b k
Hidden Layer
h 1
h 2
h k
W nk
W 12
W 1k
W n 2
W 11
W n1
Input Layer
X 1
X 2
X n