KARAR AĞAÇI KULLANARAK SALDIRI TESPİT SİSTEMLERİNİN PERFORMANS DEĞERLENDİRMESİ

Taner TUNCER **Yetkin TATAR**

Fırat Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Elazığ

ttuncer@firat.edu.tr

Özet: Karar Ağaçları gibi makine öğrenmesi teknikleri bilgisayar ağlarında saldırı tespiti için son yıllarda sıkça kullanılmaya başlanmıştır. Makine öğrenmesi teknikleri bilgisayar ağlarında normal ve anormal örüntüleri başarılı bir şekilde sınıflandırabilir. Sınıflandırma işleminin başarı testleri Yanlış Negatif Alarm Oranı (False Negative Rate) ve Doğru Pozitif Alarm Oranına (True Pozitive Rate) göre yapılır. Bilgisayar ağlarında saldırının tespiti için bu kriterden başka saldırı tespit zamanı da önemlidir. Bu makalede karar ağaçları kullanılarak saldırıların FNR, TPR ve saldırı tespit zamanı belirlenmiştir. Bu amaç için KDD Cup 99 eğitim seti kullanılmıştır. Eğitim veri setinde tanımlı 23 sınıf öncelikle 5 daha sonra 2 sınıfa indirgenerek ayrı ayrı kurallar elde edilmiştir. Saldırıların tespiti için JAVA programlama dilinde program yazılmış ve saldırıların tespit zamanı, FNR ve TPR oranları tespit edilmiştir.

1. GİRİŞ

Saldırı tespit sistemleri(STS), internet veya yerel ağdan gelebilecek ve ağdaki sistemlere zarar verebilecek, çeşitli paket ve verilerden oluşabilen saldırıları fark etmek üzere tasarlanmış sistemlerdir. Temel amaçları belirlenen kurallar çerçevesinde bu saldırıları tespit ederek mail , sms , snmp mesajları gibi araçlarla haber vermek ve gerekliyse bu saldırıyı önlemektir.

Şu anda piyasada birçok Saldırı Tespit sistemi vardır. Açık kodlu yada serbest kullanımlı ürünlerde en önemli STS SNORT'tur[1]. Açık kodlu olması, plug-in (eklenti) yazılabilmesine olanak tanıması ve desteklediği işletim sistemi platformunun fazla olması SNORT'u ağ tabanlı saldırı tespit sistemleri içerisinde öne çıkaran sebeplerdir.

STS sistemlerinde amaç doğru tespitin maksimum yanlış alarmın ve saldırı tespit zamanının minimum olmasıdır[2]. Genel olarak bir STS test edilmek istendiğinde literatürdeki DARPA ve KDD veri setleri kullanılır[3].

Bu çalışma Türkiye Bilimsel ve Teknolojik Araştırma kurumu tarafından desteklenmiştir. (TUBITAK). Proje No: 108E166

Bu veri setleri hem normal ağ trafiğini hemde saldırı trafiğini içermektedir.Normal ve anormal trafiği ayırt etmek için, Karar Ağaçları, yapay sinir ağları, genetik algoritmalar gibi makine öğrenmesi algoritmaları kullanılır.[4,5,6,7].

Saldırı tespit sistemlerinde, kötüye kullanım ve anormallik tespiti şeklinde iki yaklaşımı bulunur. Kötüye kullanım tespitinde; iyi bilinen saldırı örüntüleri kullanılarak saldırı imzaları ile eşleşen olaylar saldırı olarak belirlenir. Bu tür saldırı tespit sistemlerinde bir kural veritabanı vardır ve bu veritabanları saldırı imzaları içerir. İmza veritabanları yeni bir saldırı türü ortaya çıktığında kolayca güncellenebilirler. Kötüye kullanım tespiti yönteminde; sadece bilinen saldırılar tespit edilebilmekte, saldırı imzaları genelleştirilememekte ve fazla sayıda hatalı alarm meydana gelmektedir. Anormallik tespiti yöntemi, saldırı tespit yaklaşımlarından bir diğeridir. Bu yaklaşımın kötüye kullanım tespitine göre en büyük avantajı bilinmeyen saldırıları tespit edebilmesidir. Bu yüzden son zamanlarda birçok anormallik tespiti yöntemi geliştirilmiştir [8,9]. Anormallik tespiti sistemleri beklenen normal kullanım profillerinden sapma gösteren etkinlikleri anormallik olarak işaretlerler. Anormallik tespitinde, sistemin normal davranışı Anomaly threshold ve CUSUM gibi istatistiksel yöntemler yardımıyla elde edilir[10,11].

Bu makalede karar ağaçları kullanılarak saldırıların FNR TPR ve saldırı tespit zamanı belirlenmiştir. Bu amaç için KDD Cup 99 eğitim seti kullanılmıştır. Eğitim veri setinde tanımlı 23 sınıf öncelikle 5 daha sonra 2 sınıfa indirgenerek ayrı ayrı kurallar elde edilmiştir. Saldırıların tespiti için JAVA programlama dilinde program yazılmış ve saldırıların tespit zamanı, FNR ve TPR oranları tespit edilmiştir.

Makalenin geri kalan kısmı aşağıdaki gibi organize edilmiştir. 2. bölümde KDD Cup'99 veri seti hakkında kısa bilgi verilerek 3. Bölümde karar ağacı algoritması tanımlanmış ve kuralların nasıl elde edileceğinden bahsedilmiştir. 4. Bölümde gerçekleştirilen yazılımdan bahsedilmiştir. Son bölümde elde edilen sonuçlar detaylı olarak verilmiş ve yorumlanmıştır.

2. KDD CUP 99 VERİ SETİ

Saldırı Tespit Sistemlerinin performansını belirlemek için en zorlu aşama geçerli ve uygun veri kümelerinin elde edilmesidir. İnternet ortamından elde edilen veriler saldırının var olup olmadığına dair genel bir bilgi içermez. Saldırı için belirleyici özellik veya bilgi ağın gözlemlenmesi yoluyla elde edilebilir. Genel olarak ağın gözlemlenmesi masraflı ve gereksiz bir iş olarak görülebilir. Ancak ağ veya bilgisayar sistemlerinin çalışabilmesi için, ağdan veya bilgisayar sistemlerinden veri toplama, artık günümüzde kaçınılmaz bir süreçtir. Bu süreç biraz maliyetli olduğundan dolayı bazı ağ mühendisleri yapay veriler kullanarak ağ veya sistemlerini sorunsuz çalıştırmayı istemektedirler. Ancak yapay verinin internet trafiğine benzediğini kanıtlamak zordur. Genel olarak; Gerçek veri, saldırı türleri belli olan veri setleri bulmak ve ağ trafiğini tanımlamak ve simule etmek zordur.

Yukarıda belirtilen zorluklara rağmen saldırı tespit sistemlerini test etmek için geçerli veri kümelerine ihtiyaç vardır. Genel olarak bir ağ trafiğini bir sniffer kullanarak gözlemlenebilir. Ancak sadece ağ paketlerini gözlemlemek ağ trafiği hakkında genel bir bilgi vermeyebilir. Bu dezavantajlara rağmen saldırı tespit sistemlerinin testleri için geliştirilen birkaç veri seti bulunmaktadır.

KDD Cup'99 veri seti Saldırı tespit sistemlerinin testi için geliştirilmiş ve 5. Uluslararası Knowledge Discovery and Data Mining konferansı çerçevesinde düzenlenen bir yarışmada kullanıma sunulan veri setidir. Bu veri setindeki saldırılar 4 ana kategoride sınıflandırılabilir[3].

Denial of Service (Hizmet Engelleme): Bu saldırılar genel olarak TCP/IP protokol yapısındaki açıklardan faydalanılarak bir sunucuya birden çok bağlantı isteği göndererek yasal kullanıcıların hizmet almasını engellemeye yöneliktir.

Bilgi Tarama (Probing): Bu tür saldırılar bir sunucunun yada herhangi bir makinanın geçerli IP adreslerini, aktif portlarını veya işletim sistemini öğrenmek için geliştirilmiştir.

Yönetici Hesabı ile Yerel Oturum Açma (Remote to Local-R2L): Kullanıcı haklarına sahip olunmadığı durumda misafir yada başka bir kullanıcı olarak izinsiz erişim yapılmasıdır.

Kullanıcı Hesabının Yönetici Hesabına Yükseltilmesi (User to Root-U2R): Bu tip saldırılarda sisteme girme izni olan fakat yönetici olmayan bir kullanıcının yönetici izni gerektirecek işler yapmaya kalkışmasıdır.

KDD Cup'99 veri seti hem eğitim hemde test verisini içermektedir. Eğitim verisinde toplam 494020 örnek mevcuttur. Eğitim ve Test setindeki her bir örnek toplam 41 özellikten oluşmaktadır. Bu özellikler ise Basit, İçerik, Zaman Tabanlı Trafik ve Host Tabanlı Trafik adı altında 4 farklı grupta toplanmıştır[3].

3. KARAR AĞAÇI İLE SINIFLANDIRMA

Sınıflandırma, yeni bir nesnenin niteliklerini inceleme ve bu nesneyi önceden tanımlanmış bir sınıfa atamaktır. Burada önemli olan, her bir sınıfın özelliklerinin önceden net bir şekilde belirlenmiş olmasıdır. Sınıflandırma problemlerinde en çok kullanılan algoritmalardan biri karar ağaçlarıdır. Diğer sınıflandırma algoritmalarıyla kıyaslandığında karar ağaçlarının yapılandırılması ve anlaşılması daha kolaydır[12]. Karar ağaçları kullanılarak sınıflandırma 2 aşamada gerçekleştirilir. İlk adımda ağaç oluşturulur. İkinci adımda ise veriler tek tek ağaca uygulanarak sınıflandırma gerçekleştirilir.

Genel olarak sınıflandırma işlemi matematiksel olarak aşağıdaki gibi tanımlanır.$D=\{t_1,t_2,\ldots,t_n\}$ bir veri tabanı olsun ve her bir kayıt t_i ile temsil edilsin. $C=\{C_1,C_2\ldots C_m\}$ ise m adet sınıftan oluşan sınıflar kümesini temsil etsin. Her bir C_j ayrı bir sınıftır ve her bir sınıf kendisine ait kayıtları içerir. Yani, $C_j=\{t_i|=C_j, 1\leq i\leq n$ ve $t_i \in D\}$, dir. Veritabanındaki herbir kayıt için alanlar ise $\{A_1,A_2,\ldots A_n\}$ 'den oluşsun. Bu tanıma ilaveten her bir kayıt $C=\{C_1,C_2,\ldots C_m\}$ sınıflarından birine ait ise karar ağacı aşağıdaki gibi tanımlanır.

Her bir düğüm A_i alanı ile isimlendirilir. Kök düğüm ile yaprak arasındaki düğümler birer sınıflandırma kuralıdır. Her bir yaprağın bir sınıf olduğu ağaçtır[13].

Karar ağaçları oluşturulurken kullanılan algoritmanın ne olduğu önemlidir. Kullanılan algoritmaya göre ağacın şekli değişebilir. Değişik ağaç yapıları da farklı sınıflandırma sonuçları verebilir. Karar ağacına dayalı olarak geliştirilen algoritmalar genel olarak aşağıda verilen kaba kod çerçevesinde çalışır.

Algoritma.1 Karar Ağacı

1. **D:** Veritabanı, **T:**Ağaç
2. **T=0;** *Başlangıç Durumu Ağaç boş küme*
3. Dallanma Kriterlerini Belirle;
4. Dallanma Kriterlerine Göre T kök düğüm belirle;
5. Dallanma Kriterlerine gore Kök düğümü dallara ayır;
6. Tüm dallar için
7. **Do**
8. Düğüm oluşturmak için değişken belirle
9. **If**(Durdurma kriterine ulaşıldı)
10. Yaprak ekle, Dur
11. **Else**
12. **Return**

Karar ağaçlarına dayalı olarak geliştirilen bir çok algoritma vardır. Bu algoritmalar birbirlerinden kök, düğüm ve dallanma kriteri seçimlerinde izledikleri yol açısından ayrılırlar. Literatürde en yaygın olarak bilinen algoritmalar ID3, C4.5 ve C5dir.

C4.5 algoritması sınıflandırmada en ayırıcı özelliğe sahip değişkeni bulurken entropi kavramından yararlanır. Entropi kavramı eldeki verinin sayısallaştırılmasıdır. Entropi bir veri kümesi içindeki belirsizliği ve rastgeleliği ölçmek için kullanılır. Entropi matematiksel olarak şöyle tanımlanır.
$<p_1,p_2,\ldots p_n>$ olasılıkları ifade ederse tüm olasılıkların toplamı 1 olmalıdır. $\sum_{i=1}^{n} p_i = 1$, bu durumda entropi denklem.1' deki gibi olacaktır.

$$H(P) = \sum p_i \log(1/p_i) \qquad (1)$$

Veritabanının tamamının entropisi hesaplanır; ancak bu veritabanı farklı bölümlere ayrılırsa her bir alt bölümün de entropisi hesaplanmalıdır. Buradaki H fonksiyonu veritabanının herhangi bir durumdaki durumunu temsil etmektedir. C4.5 algoritması kullanılarak ağaç elde edilirken her bir alt ağaçlar yapraklara dönüştürülür. Ağaç yapısı oluşturmak için, herbir alt ağacın yaprağa dönüşümü denklem.2 ve denklem.3'te gösterilen kazanım ve ayırma oranları ile gerçekleştirilir.

$$KO = K(D,S) / AO(D,S) \qquad (2)$$

$$AO(D,S) = H\left(\frac{|D_1|}{|D|},\ldots,\frac{|D_s|}{|D|}\right) \qquad (3)$$

4.UYGULAMA

Bu çalışmada saldırı tespit sistemlerinden istenen tespit zamanını minimuma indirgemek için JAVA programlama dilinde bir program yazıldı. Her bir sınıf için bir thread yazılmıştır. Aşağıdaki yapı bir threadin nasıl tanımlandığı, çalıştırıldığı ve sonlandırıldığını göstermektedir. Bu amaç için öncelikle KDD Cup' 99 eğitim seti Karar ağacı algoritmalarından biri olan C4.5 kullanılarak sınıflandırılmıştır. Sınıflandırma işlemi sonucunda her bir saldırı sınıfına ait kurallar elde edilmiştir. Elde edilen kurallar ile saldırıların sınıflandırma işlemi yapılmış sonuçta test örneklerinin ne kadarlık bir zaman süresinde sınıflandırıldığı tespit edilmiştir. Daha sonra eğitim setindeki sınıf sayısı 23' ten 5' e son olarak ise 2 sınıfa indirgenmiştir. Her bir durum için kural sayısının, tespit zamanının değişimi, TPR ve FNR değişimi elde edilmiştir.

Algoritma.2 Thread Yapısı

```
class [thread adı] extends Thread{
  public void run(){
....
....
      }
      }
public class [Sınıf adı] {
  public statc void main(argümanlar){
  [thread name]thread1=new[thread adı()];
  [thread name] thread2=new [thread adı()];
  thread1=start();
  thread2=start();
  }
}
```

Her bir thread ilgili sınıfa ait kuralları içermektedir. Tüm sınıflar için kurallar kullanım yüzdesine göre sıralanarak thread oluşturulmuştur. Diğer bir deyişle en sık kullanılan kuraldan en az kullanılan kurallara göre sıralaması yapılmıştır. Test verilerinin kurallar ile karşılaştırılması en çoktan en aza doğru yapılmıştır. Dolayısı ile tespit zamanı minimuma indirgenmeye çalışılmıştır. Kuralın eğitim setinde tespit ettiği örnek sayısı, N, Toplam örnek sayısı T olmak üzere, kural kullanım yüzdesi aşağıdaki denklem.4 ile verilir.

$$K\% = \frac{N}{T} \qquad (4)$$

311029 test verisinin sınıfının belirlenmesi için test verisi kurallar ile karşılaştırılarak tespit zamanı hesap edilmiştir. 10 farklı test yapılarak tespit zamanları elde edildikten sonra program sonlandırılmıştır. Her bir test yapılırken bazı test verileri için tespit zamanları işlemcinin yoğun çalışmasından dolayı farklıdır. Dolayısıyla aşırı pik değerleri içeren zaman süreleri vardır. Ortalama tespit zamanını belirlemek için piklere sahip zaman sürelerinin elenmesi gereklidir. Bu amaç için z testi kullanılmıştır. z değeri genel olarak test uygulamalarında 2.5 birimdir. Tüm örneklerin tespit edilme zamanlarının ortalama değeri M, i. Örneğin tespit edilme zamanı X_i, ve standart sapması σ ise z aşağıdaki gibi verilir.

$$z = \frac{|M - X_i|}{\sigma} \qquad (5)$$

Pik zaman değerlerine sahip değerler elendikten sonra elde edilen zaman serisinin ortalaması alınarak tespit zamanı belirlenmiştir. Saldırı tespitinde diğer önemli kriter FNR ve TPR oranıdır. Saldırı tespit sistemlerinde istenen FNR oranının minimum TPR oranının maksimum olmasıdır. Yanlış alarm oranı büyüdükçe doğru tespit oranı azalır. FNR ve TPR tanımını yapabilmek için sınıflandırma başarımını incelemede yarar sağlayan Confusion matrisi aşağıdaki gibi tanımlanır.

Tablo.1 Confusion Matris

		Tahmin Sınıf	
		Evet	Hayır
Doğru Sınıf	Evet	TP	FN
	Hayır	FP	TN

$$FNR = \frac{FN}{TP + FN} \qquad (6)$$

$$TPR = \frac{TP}{TP + FN} \qquad (7)$$

5. SONUÇLAR

Testler PIV 3.0Ghz işlemcili 512 MB RAM'e sahip desktop üzerinde yapılmıştır. Tablo.2 23 sınıfa ait kural sayısı FNR ve TPR başarımlarını göstermektedir. Tablo.3 5 sınıfa ait kural sayısı, FNR ve TPR başarımlarını gösterirken Tablo.4 2 sınıfa ait kural sayısı, FNR ve TPR başarımlarını göstermektedir. Yukarıda bahsedilen thread yapısı kullanılarak gerçekleştirilen yazılım ile saldırı tespitinde kullanılan herhangi bir kuralın, ortalama tespit zamanı değişimi elde edilmiştir. Şekil.1 ortalama tespit zamanının sınıf sayılarına göre değişimini göstermektedir. Sınıf sayılarının azaltılması ile saldırılar için elde edilen kurallar sayısı toplamda azaltılmıştır. Normal sınıfına ait kural sayısında artış gözlemlenmiştir. Ancak toplam kural sayısının artmasına rağmen ortalama saldırı tespit zamanında belirgin bir şekilde düşüş gözlemlenmiştir. 23, 5 ve 2 sınıflandırma için FNR ve TPR değişimlerinde ise göze çarpan bir değişiklik olmamıştır.

Tablo.2 23 Sınıf için Kural sayısı FNR veTPR

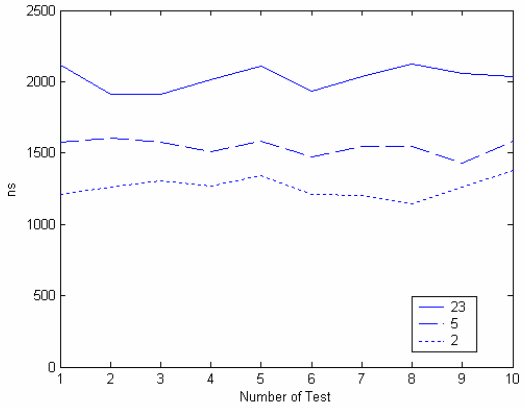
Attack Class	Kural Sayısı	FNR	TPR
Smurf	1	7.10^{-6}	0,999
Neptune	7	10^{-6}	0.999
Back	6	0,004	0,995
Teardrop	1	0	1
Pod	1	0,018	0,981
Land	2	0	0,952
Satan	6	0,009	0,99
Ipsweep	9	0,002	0.997
Portsweep	6	0,025	0,974
Nmap	3	0,041	0,958
Warezclient	9	0,009	0,99
Guess_Passwd	3	0,036	0,963
Warezmaster	2	0,1	0,9
Imap	2	0,142	0,857
Ftp_write	3	0,384	0,615
Multihop	2	0,333	0,666
Phf	1	0	1
Spy	1	0,5	0,5
Buffer_overflow	4	0,071	0,928
Rootkit	3	0,474	0,526
Loadmodule	3	0,437	0,562
Perl	1	0	1
Normal	29	$1,5.10^{-4}$	0,999

Tablo.3 5 sınıf için Kural sayısı, FNR ve TPR

Attack Class	Kural Sayısı	FNR	TPR
Dos	18	$4,8.10^{-5}$	0,999
Probe	16	0,013	0,986
R2L	15	0,02	0,978
U2R	4	0,413	0,586
Normal	36	$1,6.10^{-4}$	0,999

Tablo.4 2 sınıf için Kural sayısı, FNR ve TPR

Attack Class	Kural Sayısı	FNR	TPR
Abnormal	37	3.10^{-4}	0,999
Normal	39	10^{-4}	0,999



Şekil.1 23,5 ve 2 Sınıf için Saldırı tespit zamanı değişimi

KAYNAKLAR

[1] Caswell, B., M. Roesch , Snort: The open source network intrusion detection system. http://www.snort.org/.ddd, 2004

[2] Denning, D. E. , An intrusion-detection model. IEEE Transactions on Software Engineering 13 (2), 222–232, 1987.

[3] KDD Cup 1999 data. [Online]. Available: http://www.kdd.ics.uci.edu/databases/kddcup99/kddcup-99.html. 1999.

[4] Sheen, Shina Rajesh, R. , "Network intrusion detection using feature selection and Decision tree classifier" TENCON 2008 - 2008, TENCON 2008. IEEE Region 10 Conference, pp:1-4, 2008.

[5] Tarek A., Adel B., Michael R., "Protocol analysis in intrusion detection using decision tree" Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2, 2004.

[6] Shun, J. Malki, H.A., "Network Intrusion Detection System Using Neural Networks", ICNC '08. Fourth International Conference on, Volume: 5, pp: 242-246, 2008

[7] Jiu-Ling Z., Jiu-Fen Z., Jian-Jun L., "Intrusion detection based on clustering genetic algorithm", Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on Volume: 6, pp: 3911- 3914 Vol. 6, 2005.

[8] Marina T., Chuanyi J., "Anomaly Detection in IP Networks", IEEE Transaction on signal processing, Vol. 51, No. 8,2003.

[9] Dan L., Kefei W., Jitender S. D., "FADS: Fuzzy Anomaly Detection System", LNAI 4062, pp:792-798, 2006.

[10] Vasilios A. Siris, Fotini P., “Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks”, Computer Communications, pp:1433-1442, 2006.

[11] T. Tuncer, Y. Tatar, "Detection SYN Flooding Attacks Using Fuzzy Logic", International Conference on Information Security and Assurance, pp:321-325 2008.

[12] Agrawal R., “Database Mining : A Performance Perspective”, IEEE Transactions on Knowledge and Data Engineering, pp:914-925,1993

[13] Dunham M.H., “ Data Mining Introductory and Advanced Topics”, Prentice Hall, Pearson Education Inc., 2003.