Proceedings of the International Conference on All Aspects of Cyber Security 2019 (A2CS'19)

Makine Öğrenmesi ile İşletim Sistemi Tarama Saldırısı Tespiti

Operating System Scan Attack Detection with Machine Learning

Rabia DOĞAN
Yönetim Bilişim Sistemleri Bölümü
Pamukkale Üniversitesi
Denizli, Türkiye
rdogan171@posta.pau.edu.tr

Ebu Yusuf GÜVEN
Bilgisayar Mühendisliği Bölümü
İstanbul Üniversitesi-Cerrahpaşa
İstanbul, Türkiye
eyguven@istanbul.edu.tr

Muhammed Ali AYDIN
Bilgisayar Mühendisliği Bölümü
İstanbul Üniversitesi-Cerrahpaşa
İstanbul, Türkiye
aydinali@istanbul.edu.tr

***Özetçe* — Siber güvenlik, artan siber tehditlerle kurum ve kuruluşların en önemli konusu haline gelmektedir. Yapay zekâ yöntemleri kullanılarak bu siber tehditlerin tespit edilmesi mümkündür. Siber saldırıları tanımlayarak belirli aşamalarda tekrar edilen yöntemler üzerinden ağ davranış analizi çıkarılması farklı saldırganların ve yöntemlerin tespit edilmesine imkân sağlamaktadır. Siber saldırıları henüz keşif aşamasında tespit etmek, siber saldırıların en az zararla atlatılması için önemlidir. Yapılan bu çalışmada Kitsune veri setinin Os scan keşif saldırısı bölümü kullanılmıştır. Veri setinden alınan örneklem üzerinde Karar Ağacı, Random Forest, Destek Vektör Makinesi, K en yakın komşu, Naive Bayes ve Adaboosting algoritmaları kullanılarak algoritmaların doğruluk ve performans parametreleri karşılaştırılarak sonuçlar paylaşılmıştır.**

Anahtar Kelime—Siber Saldırı; Makine Öğrenmesi,;İşletim Sistemi Tarama Saldırısı

***Abstract*— Cyber security, become one of the most important issue in terms of establishment and institutions with increasing rate of cyber attacks. With using artificial intelligence methods is possible to be determined of cyber threats. By identifying cyber attacks, it is possible to identify different attackers and methods by conducting network behavior analysis through repetitive methods at certain stages. It is important to detect cyber attacks during the discovery process to avoid cyber attacks with minimal damage. In this study, OS scan attack section of Kitsune dataset is used. The results were shared by comparing the accuracy and performance parameters of the algorithms using Decision Tree, Random Forest, Support Vector Machine, K Nearest Neighbor, Naive Bayes and Adaboosting algorithms.**

Keywords—Cyber Attack; Machine Learning; OS Scan Attack

I. GİRİŞ

Siber güvenlik gelişen teknolojiyle birlikte kurum ve kuruluşların hatta hükümetlerin en önemli gereksinimi haline gelmiştir. Bu nedenle siber güvenliğin sağlanmasında birçok ürün ve hizmet geliştirilmekte ve saldırılar tespit edilebilmektedir. Siber güvenlik için geliştirilen ürün ve hizmetler başlangıçta kural tabanlı sistemler kullanılarak geliştirilmiş olmasına karşın yapay sinir ağları ve makine öğrenmesi gibi tekniklerin gelişmesiyle davranış temelli saldırı tespit sistemleri geliştirmek mümkün hale gelmiştir [1]. Bu yapay zekâ yöntemleri sayesinde, kural tabanlı sistemleri atlatan saldırganlar dahi güvenlik sistemleri tarafından tespit edilebilmektedir [2]. Bunun için benzer ağ davranışları yapay zekâ kullanılan saldırı tespit sistemlerine öğretilmesi gerekmektedir [1-5]. Ayrıca öğrenme aşaması olmadan, denetimsiz öğrenme yöntemleri ile ağ trafiğinin sınıflandırılması mümkündür [1] [6].

Siber saldırganlar sistemlere saldırmadan önce sistemi tanımak için bir dizi keşif saldırıları yaparlar. Bu keşif saldırıları ağ topolojisini çıkarmak ve ağa bağlı cihazları analiz etmek için kullanılmaktadır. Bu sayede saldırganlar ağ üzerinde yapacakları saldırılara karar verirler. IP Scan ağ topolojisi için, OS Scan cihazın işletim sisteminin Port Scan cihaz üzerinde aktif açık portların ve çalışan servislerin tespiti için kullanılmaktadır. Bu çalışma kapsamında özellikle işletim sistemi tespit etme saldırılarının tespit edilmesi için makine öğrenmesi yöntemleriyle model geliştirilmiştir.

Daha önce yapılmış olan çalışmalarda, siber saldırganların ağ üzerindeki davranışlarının analiz edilerek normal trafik ile saldırı trafiğinin ayrılması çalışmaları yapılmıştır [1, 3, 7]. Bu ayrım tipik ikili sınıflandırma problemi olarak karşımıza çıkmaktadır. Karar destek makinesi (SVM) [8,9], Karar Ağacı [10,11], Naive Bayes [8,12], Adaboosting [13], Random Forest [14,15] ve K En Yakın komşu(KNN)[16] algoritmaları bu tür saldırıların tespiti için sıklıkla kullanılmaktadır. Os Scan saldırısı tespiti için yapılan çalışmalardan literatür bölümünde genişçe bahsedilmiştir.

Bu çalışmada farklı makine öğrenmesi teknikleri bir arada kullanılarak ağ trafiğinin sınıflandırması yapılmaktadır. Os Scan saldırısı ağ üzerinde sıklıkla kullanılan ICMP/Ping paketleri kullanılarak yapılması nedeniyle, yüksek doğrulukta sınıflandırma yapılması zor keşif saldırısı yöntemlerindendir [7,17]. Bu nedenle yüksek doğrulukta tespit edilmesi kural tabanlı yöntemlere karşı zordur. Deneysel olarak hazırlanmış saldırı senaryolarının ağ hareketlerini ve paketlerini 115 özelliğe dönüştüren Kitsune [1] veri setinin OS Scan bölümünden alınan bootstrapping örneklemi üzerinde çalışılarak sonuçlar paylaşılmıştır.

Bu yayında 2. bölümde siber saldırıların ve özellikle Os scan saldırısını tespit etmek için kullanılan yapay zeka yöntemlerine değinilirken 3. Bölümde Keşif saldırılarından os scan ve sıklıkla kullanılan NMAP uygulamasına yer verilmiştir. 4. Bölümde kullanılan makine öğrenmesi yöntemleri açıklanırken, 5. Bölümde bu yöntemlerin sonuçlarıyla ilgili bulgular paylaşılmıştır. Son olarak da 6. Bölümde sonuç ve tartışmanın yanında yapılacak işler açıklanmaktadır.

II. LİTERATÜR

Y. Mirsky ve arkadaşları, autoencoders kullanılarak Mirai, Os Scan, ARP MitM, SSDP Flood, Aktif Wiretap, SYN Flooding, Fuzzing ve Video Enjeksiyonu saldırıları düşük güç tüketen cihazlar üzerinden denetimsiz öğrenme yaparak tespit etmeye çalışmaktadır. Kİtsune adında bir veri seti yayınlamanın yanında ağları korumak için kullanılan bir ağ saldırı tespit sistemi (NIDS) tanıtılmıştır. Kitsune'un çekirdek algoritması KitNet autoencoders (sinir ağları) kullanarak çevrimiçi anomali tespiti yapmaktadır [1].

S.C. Lee ve D.V. Heinbuch tarafından, sinir ağları yapısından oluşan ve anomali tespiti yapabilen bir saldırı tespit sistemi (IDS) denemesi anlatılmıştır. Deneysel bir çalışma yapılmış olup, bu çalışmalarda birçok saldırı yöntemi ile protokol veya işletim sistemine saldırı gerçekleştirilmiştir. Eğitilen yapay sinir ağları (ANN), öğrenme sürecinde kullanılmayan saldırı verileri ile test edildiğinde saldırıyı tanıyabilmektedir. Kullanılan bazı yöntemler sadece anomali tespiti için değil, saldırıları sınıflandırmak için de kullanılabilir. Geliştirilen IDS'de saldırı tespiti için geri yayılma sinir ağları (Back Propagation Neural Network) kullanılmış ve bir ağ simülasyonu tarafından üretilen test verileri kullanılarak değerlendirilmiştir [2].

Saldırı tespit sistemlerinin büyük çoğunluğu imza tabanlı teknolojilere dayanmaktadır. T. H. Ptacek ve arkadaşı saldırı tespiti için bir imza tanıma (signature recognition) paradigması tanıtmıştır [9]. Çalışmada saldırı davranışlarını normal kullanımdan ayırmak için çeşitli veri kaynakları ve örüntü tanıma teknikleri kullanılır. Bu sistemlerde saldırı kontrolü performans için çalıştıkları imza karşılaştırma ve kayıtlı veri tabanı ile sınırlıdır. Tüm varyasyonlar veri tabanında bulunmuyor ise bilinen saldırılar bile gözden kaçırılabilir [9].

Guo Xiaobing ve arkadaşları, mevcut ağ tarama yöntemlerinden ICMP echo scan, TCP connect scan, TCP SYN scan, TCP FIN scan, TCP ACK scan, TCP reverse-ident scan ve UDP scan saldırısını sınıflandırılmıştır. Ayrıca bu saldırılar analiz edilmiş, ağ taramasında iyi bir tespit ve koruma stratejisi sunulmuş ve çalışmanın performansı gösterilmiştir. IEDP (Impartation and Error-steering Detection and Protection system) adında tespit ve koruma sistemi saldırı senaryoları üzerinden anlatılmıştır. Bir tarama tespit edilirken, IEDP tarayıcıyı kandırmak ve kafasını karıştırmak içim sahte bilgi verir. Bu nedenle portları tararken, tarayıcı tüm portların (bağlantı noktalarının) dinlendiğini ve hangi portun gerçek olduğunu söylememektedir. Tarayıcı çok fazla sahte bilgi edinir ve hangisinin gerçeği olduğunu söyleyemez, bu da taramada başarısızlığa neden olur. IEDP, tarayıcıyı daha etkili bir şekilde şaşırtmak için hataları sahte paketlere yönlendirir. Ayrıca IEDP taramayı kandırmak için "error steeling" adında yeni bir mekanizma tanıtılmıştır. IEDP tarayıcı ile iletişimdeki hataları rastgele yönlendirir ve tarayıcıda iletişimin dengesiz olduğuna ve taramayı bıraktığına inanmasını sağlar. IEDP tespit ve koruma sistemi, ICMP echo scan, TCP Connect Scan, TCP SYN Scan, TCPACK Scan, TCPFIN Scan ve UDP Scan teknikleri ile test edilmiştir [17].

Jayant Gadge ve Anish Anand Patil, port tarama girişimlerini tespit etmeyi ve port tarama girişimlerinin yapıldığı makine hakkında bilgi sunmayı amaçlamıştır. Kullanılan işletim sistemi, saldırının (taramanın) geldiği yerin olası konumu, WHOIS veri tabanından ve Traceroute'tan gelen bilgiler tarayıcı hakkında bilgi sağlamada yardımcı olmaktadır. Hem Windows hem de Linux Nmap portu, Angry IP ve MegaPing gibi diğer port tarayıcılarla birlikte kullanılan test ortamında yapılan çalışmanın test edebildiği belirtilmiştir. Tarayıcı hakkında ek bilgi almak, tarayıcının kimliğini ortaya çıkarmak için yardımcı olacaktır. Hangi portların tarandığını anlayarak, ne tür bir saldırı izleyebileceğini tahmin etmek mümkündür. Bu, Yöneticinin bir saldırı olmadan önce tedbir almasına yardımcı olur [7].

Abhay Nath Singh ve R.C. Joshi, ağ saldırı trafiğinin verimli olarak yakalanması ve analizi için bir honeypot sistemi tanıtılmıştır. Honeypot saldırganları ve bilgisayar korsanlarını kritik kaynaklardan uzaklaştırmanın yanında, saldırganın yöntemlerini ve araçlarını tespit etmek için kullanılan bir bilgi sistemi kaynağıdır. Honeypots oluşturmak için en yaygın kullanılan araçlardan biri de honeyd'dir. Önerilen sistem, kayıt ve analiz modülünden oluşmaktadır. Kayıt modülü, kayıt boyutunu küçülterek disk alanından tasarruf sağlamaktadır. Honeyd tarafından Faked OS personality, Faked network topology, Faked TCP/IP stack, Faked services ve Faked buffer overflow saldırıları izlenmiştir. Sahte OS (işletim sistemi) kişiliği (Faked OS personality), gerçek bir işletim sistemi gibi davranır, görünür ve işletim sistemini başarıyla taklit eder. Ayrıca belirlenen işletim sistemi kişilikleri statik olarak belirlenmiştir [3].

Cynthia Bailey Lee ve arkadaşları tarafından yazılan bu makalede port taramalarının özelliklerini keşfetmek ve sınıflandırmak için örnek ağ izleri analiz edilmiştir. Projede, port tarama trafiği analiz edilir ve karakterize edilir. Port tarama tespit mekanizması geliştirilmiştir ve genellikle izinsiz giriş tespit sisteminin bir parçası olarak sisteme dahil edilmiştir. Bu sezgisel tarama Spice operasyonel olarak pratik olmasına rağmen gizli port taramalarını tespit etmesini sağlamaktadır. Port taramaları Vertical Scans (dikey), Horizontal Scans (yatay) ve Block Scans (blok) olarak üç temel türe ayrılmıştır. Daha

gelişmiş sezgisel tarama ve sınıflandırma, port tarama trafiğinin daha doğru tanımlanmasını sağlaması hedeflenmektedir [6].

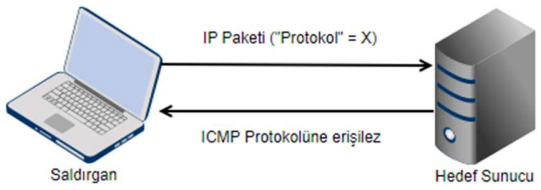
Gregory Conti ve Kulsoom Abdullah, saldırganlar tarafından kullanılan belirli metodolojiler ile çeşitli popüler ağ saldırı araçlarının bıraktığı çarpıcı görsel parmak izlerini ve çeşitli görselleştirme tekniklerinin ağa izinsiz giriş tespitine uygulandıklarındaki güçlü ve zayıf yönleri sistematik olarak incelemiştir. Kullanılan teknikler tamamen pasiftir ve neredeyse hiç tespit edilemez. Tipik otomatik izinsiz giriş tespit sisteminin imzaları ve birkaç görselleştirme tekniğinin uygulanması araştırılmıştır. Bu görselleştirmeler birçok popüler ağ güvenliği aracı kullanılarak test edilmiştir. Bazı gizli saldırılar, geleneksel izinsiz giriş algılama sistemleri tarafından algılanmaya karşı dayanıklıdır, ancak uygun görselleştirme teknikleri kullanılarak kolayca görülebilir. Görselleştirme teknikleri, yaklaşmakta olan saldırılar için ipuçları sağlayabilir ve hızlı yanıt analizini kolaylaştırabilir. Klasik algoritmik saldırı tespit sistemleri (IDS) izinsiz girişleri keşfetmek için makine tarafından algılanan imzalara ve istatistiksel anormalliklere dayanır. Passive promiscuous, mode sniffing ve doğru görselleştirmeler tarafından popüler saldırı araçlarının ağ keşfi ve zayıflık araştırma sınıfları kolayca tespit edilebilmektedir [4].

Roya Ensafi ve arkadaşları, bir saldırgan, kurban ve zombi için bir ağ protokol yığınının geçiş sistemi modelini oluşturmuş ve bu modeli model denetimi kullanarak parazit olmayan özellikler için test etmenin sonuçları sunmuştur. Her ikisi de uygulama yoluyla doğrulanmış olan modellemenin sonucu olarak iki yeni idle port scanning sunulmuştur. Idle port scanning, gizlice bir kurbanın IP adresini almak ve bir portun açık olup olmadığını denetlemek için ya da zombi ve kurban arasındaki IP tabanlı güven ilişkisinden sonuç çıkarmak amacıyla zombie host taramalarını atlatmak için yan-kanal ataklarını kullanır. Idle Port taramalarından biri TCP RST hız sınırlamasına, diğeri SYN önbelleklerine dayanmaktadır [18].

III. KEŞİF SALDIRILARI

Keşif saldırıları bir saldırının başlangıç aşaması olarak görülebilir. Saldırı yapmadan önce ağı analiz etmek ve ağdaki cihazları tanımlamak için kullanılan sistematik bir girişimdir [19]. Saldırgan çeşitli tarama metotlarını kullanarak hedef ağdaki IP adreslerini belirler ve daha sonra ağdaki cihazlara özel port taraması gibi işlemlere başlar. Sistemde fark edilmeden dışarısı ile iletişimde bulunmak, hedef sistemler hakkında bilgiler elde etmek ve bu sistemlere etkin saldırılar düzenlemek keşif saldırılarının hedefleri arasındadır.

Keşif saldırısı yapılırken ilk aşama ağa bağlı cihazların tespitinin yapılmasıdır. Bu tarama iki şekilde yapılır. İlk olarak taramada ICMP/Ping paketleri tek tek sunucu IP adreslerine gönderilir veya ICMP yayınlanır ve tüm ağın yayın adresine gönderilir. Keşif saldırısının ikinci aşaması ağ topolojisi ve güvenlik duvarı kurallarının tespit edilmesidir. Güvenlik duvarı tespiti iki aşamada gerçekleştirilir. **Şekil 1** de görüldüğü gibi önce saldırgan ve güvenlik duvarı arasındaki sunucu sayısı tespit edilir, daha sonraki aşamada ise arada kalan sunucu sayısından 1 büyük TTL (time to live) değeri belirlenir, bu değere sahip portlara TCP/UDP segmentleri gönderilir, bu mesajı geri döndüren segment portlarının açık olduğu anlaşılır. Hedef sunucuda çalışan servisler için Port taraması yapılırken, IP paketi seçilen protokol ile gönderilir. Port Unreachable mesajı geri dönülmemişse açık olduğu anlaşılır. Tarama sonucunda portlar açık, kapalı ya da filitrelenmiş olarak sınıflandırılır.



Şekil. 1.İşletim sistemi tespit etme adımları

Keşif saldırıları sistemin güvenliğini test etmek için yapılan Pentest aşamalarında da kullanılmaktadır. Pentest, test kapsamındaki mantıksal hataları ve zafiyetleri tespit ederek sistemdeki hata ve zafiyetlerin istismar edilip kötü niyetli kişiler tarafından lehine kullanımını önlemek amaçlı yapılan test aktiviteleridir. Pentest aktiviteleri bir ağ erişimi ile başlar. Penetration test aktiviteleri ele geçirilen sistemlere saldırı amaçlı araçların yüklenmesini, sistemlere karşı sistem bütünlüğünü bozabilecek ve hatta sistemlerde hizmet kesintine yol açabilecek müdahalelerin yapılmasını içerir. Bu durum sistem bütünlüğü ve hizmet sürekliliği açısından risk doğurur. Bu risk kaçınılmazdır, ancak test zamanlaması, sistem yöneticileri ile koordinasyon, kimi durumlarda test öncesinde yedek alınması ve testi yapan kişilerin yetkinlik düzeylerinin yüksek olması gibi faktörlerle risk azaltılmaya çalışılmalıdır. Pentest bilgi toplama, tarama ve sınıflandırma, erişimi elde etmek, erişimi yönetmek ve izleri gizleme olarak 5 adımdan oluşmaktadır.

A. İşletim Sistemi Tarama Saldırıları (OS SCAN Attack)

Ağ taraması ağa saldırmanın ön şartıdır. Başarılı bir saldırı için tarama yapmak gerekir. Ağ taramasının amacı, ana bilgisayarın veya ağın bilgisini elde etmektir. En güçlü tarama örneklerinden biri, birçok güvenlik açıklığının işletim sistemi (OS) sürümüne bağlı olmasıdır. İyi bir TCP / IP tarayıcı ile bilgisayar korsanları hangi işletim sistemi sunucusunun çalıştığını hızlı bir şekilde bulur ve amaçlarına ulaşmak için Shell kodlarını ayarlayabilir [17].

İşletim sistemi tarama yöntemleri ise aktif tanıma ve pasit tanıma olarak sınıflandırılır. Aktif tanımada paketler uzak bir ana bilgisayara gönderilir ve gelen cevaplar analiz edilir. Pasif tanımada ise paketler bir ağdaki ana bilgisayardan analiz edilir. Hedef sunucunun ağ iletişimi dinlenir fakat etkileşim yoktur. Aktif tarama için Nmap, Pasif tarama için NetworkMiner ve Satori adlı araçlar kullanılabilir. Pasif olarak yakalanan paketleri gerçek zamanlı olarak etkili bir şekilde görselleştirecek görselleştirmeler tasarlandığında işletim sistemine ilişkin öngörü sağlanabilir [4].

Nmap'in en çok bilinen özelliklerinden biri de uzaktan işletim sistemi algılamasıdır. İşletim sistemi tanındığında saldırgan hangi yöntemlerle sisteme sızacağına karar verebilir. Nmap

bunu açık olan port numaralarına ve gelen paketlerdeki fingerprinting bakarak bulmaya çalışır. Makinede çalışan işletim sistemi biliniyorsa fakat Nmap makinenin işletim sistemini tahmin edemiyorsa ve koşullar iyiyse (en az bir açık port ve bir kapalı port bulunmuşsa), Nmap fingerprintini girebileceğiniz bir URL sağlar.

İşletim sistemi tespiti yapmanın sebepleri,

- Hedef ana bilgisayarın güvenlik açığını belirleme,
- Açıkları kapatma (tailoring exploits),
- Ağ envanteri ve desteği (Network inventory and support),
- Yetkisiz ve tehlikeli cihazların tespiti olarak sıralanabilir.

Web tarayıcı bir site ziyaret edildiğinde, bu siteye "user agent" adı verilen bir miktar bilgi gönderir. User agent, tarayıcı, platform ve işletim sistemi sürümü hakkında bilgilere sahiptir. Bu bilgiler kullanılarak, hangi İşletim Sistemi kullanıldığı hakkında bir tahminde bulunulabilir.

İşletim sistemi algılama işlemi gerçekleştirilirken kullanılan 5 ayrı yöntem vardır. Bunlar Sequence Generation, ICMP Echo, TCP Explicit Congestion Notification, TCP ve UDP'dir. Bu yöntemlerin her biri bir veya daha fazla paketten oluşur. Her pakete verilen cevap işletim sistemi tipini belirlemeye yardımcı olur.

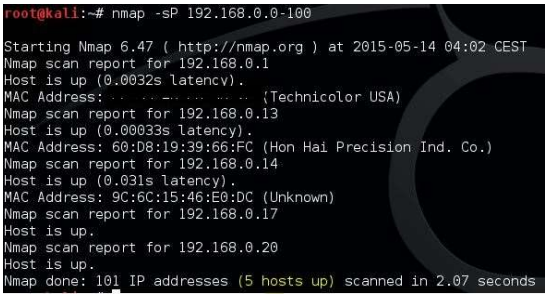
Sequence Generation yöntemi 6 TCP SYN paketinden oluşur. Her bir TCP SYN paketinin sonucu NMAP'ın OS tipini belirlemesine yardımcı olmaktadır. İki ICMP Request paketi değişen ayarlarla birlikte sisteme gönderilir ve ortaya çıkan yanıtlar OS tipinin NMAP ile doğrulanmasına yardımcı olmaktadır. TCP Explicit Congestion Notification çok fazla paket üretildiğinde sistemler tıkanıklığı azaltmak için yavaşlar. Gönderilen paket yalnızca hedef sistemden bir yanıt almak içindir. Dönen belirli değerler, belirli bir işletim sistemini belirlemek için kullanılır. Çünkü her işletim sistemi paketleri farklı şekillerde ele alır. TCP'de paketler belirli ayarlar ile gönderilir, sonuçlar işletim sistemine bağlı olarak değişmektedir. UDP ise kapalı bir porta gönderilen tek bir paketten oluşur. Eğer hedef sistemde kullanılan bağlantı noktası kapalı ise güvenlik duvarı yoktur.

Zafiyetli cihazlar bulmak için kullanılan yöntemlerden biri de port taramasıdır. Port taraması, davetsiz misafirlerin bilgisayarlara erişim sağladığı açık kapılar olarak tanımlanabilir. [20] Bu teknik bağlantı noktasına mesaj gönderme ve yanıtı dinleme işleminden oluşur. Alınan yanıt, port durumun ve ana bilgisayarın işletim sistemi hakkında bilgiler verir. Bir sistemin zayıf yönlerini ve bir saldırı için en iyi yöntemi belirleyen bir çok araç vardır. Bu araçlardan Nmap, host makinesinin işletim sistemini bulmak için aktif tarama (probing) teknikleri kullanır ve paket probu(probe) seçeneklerini değiştirir.

B. NMAP Uygulaması

Nmap sızma testlerinde sık kullanılan araçlardan olmakla birlikte tarama yaparken diğer bilgisayarın portlarını hedef alır. Bilgisayarların açık portlarının belirlenmesi ve çalışan servislerin tespiti için kullanılır. Nmap taraması yapıldığında ağa bağlı makinenin işletim sistemi, çalışma süresi, yazılımların hangi servisleri kullandığı, bilgisayarın güvenlik duvarına sahip olup olmadığı ve ağ kartıyla ilgili bilgiler elde edilir.

Pentesting'de nmap yazılımı kullanıldığında ilk önce IP adresi öğrenilir. Sonra nmap taraması yapılır. Nmap sızma testleri sürerken kullanıcıya yardımcı olan bir araçtır. Nmap tarama yapılan ağın haritasını oluşturabilir, hangi portların açık veya filtreli olduğunu listeyebilir. **Şekil 2** de NMAP uygulamasının konsol üzernden çalıştırlması görülmektedir



Şekil. 2.N Map Tarama Yaparken Ekran Görüntüsü

IV. YÖNTEM

A. Ensemble Method

Ensemble learning/methods, birkaç makine öğrenmesi tekniğini bir modelde birleştiren meta algoritmalardır. Birçok ensemble method, tek temel öğrenme algoritması kullanır. Böylece daha iyi tahmin performansı elde edilir.Ensemble yöntemler sequential ensemble methods ve parallel ensemble methods olarak iki gruba ayrılabilir.Temel öğrenicilerin sırayla üretildiği sıralı topluluk yöntemlerine örnek olarak AdaBoost, paralel olarak üretildiği paralel topluluk yöntemlerine örnek olarak Random Forest söylenebilir.

B. Naïve Bayes

Naive Bayes Sınıflandırma Modeli Bayes Teoreminin bağımsızlık koşulu ile basitleştirilmiş halidir. Bayes teoremi aşağıda verilen denklem 1 de ifade edilir;

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \tag{1}$$

P($A|B$) ; B olayı gerçekleştiğinde A olayının meydana gelme olasılığıdır (koşullu olasılık)

P($B|A$) ; A olayı gerçekleştiğinde B olayının meydana gelme olasılığıdır

P(A) ve P(B) ; A ve B olaylarının olasılıkları

Naive Bayes; Bayes teoremini temel alan olasılıkçı bir sınıflandırma tekniğidir. Öğrenilmek istenen kavramın tüm bu özelliklere koşulsal bağlı olduğu bir Bayes ağı olarak düşünülmektedir. Sınıflandırma yaparken her bir tanımlayıcı öznitelik birbirinden bağımsız olarak değerlendirir.

Naive Bayes Sınıflandırma Modeli, birçok özellik ve bir hedef değişkeninden oluşur.

$$p(C|F_1,...F_n) = \frac{P(C)p(F_1,...F_n|C)}{p(F_1,...F_n)} \quad (2)$$

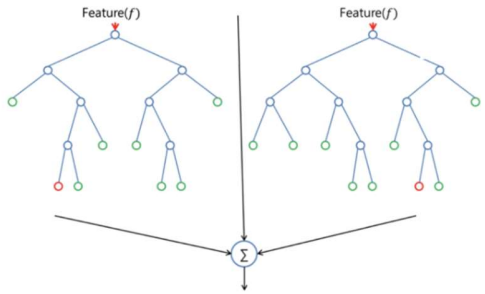
Denklem 2'de verilen Naive Bayes Sınıflandırma Modeli formülünde C verilen hedefi, F özellikleri temsil etmektedir. Bütün koşullu olasılıkların çarpımı yöntemi Navie Bayes Sınıflandırıcının elde edilmesine fayda sağlamaktadır.

C. Random Forest

Random Forest, esnek ve kullanımı kolay, denetimli bir makine öğrenmesi sınıflandırma algoritmasıdır (Supervised classification algorithm). Bu algoritma mevcut makine öğrenmesi sistemlerini oluşturan hem basitlik hem de sınıflandırma ve regresyon problemlerinde kullanılabilir. Kurulan sınıflama ve regresyon karar ağaçları bizi sonuca götürecek karar ormanı topluluğunu oluşturur ve karar ormanı oluşumu sırasında elde edilen sonuçlar bir araya getirilerek en son tahminleme yapılır.

Random Forest algoritması sınıflandırma problemlerindeki aşırı uyum problemini önlemektedir ve eğitilmiş veri setinde bulunan özellikler arasındaki en önemli özelliği tanımlamak için kullanılabilir.

Random Forest yöntemi, rastgele bir orman oluşturur. Kurduğu orman çoğu zaman bagging yöntemi ile eğitilmiş karar ağaçları topluluğundan oluşur. Karar ağaçları topluluğu veri setinden rastgele seçilmiş alt kümelerdir. Bagging yönteminin amacı öğrenme modelleri kombinasyonunun değerini yükselterek daha doğru ve istikrarlı bir tahmin elde etmek için onları birleştirmektir. Random Forest'ın iki ağaç olarak **Şekil 3**'de görülmektedir.



Şekil. 3.İki ağaçtan oluşan Random Forest

Random Forest iyi bir tahmin geçerliliği ve model yorumlanabilirliği sağlar. Bu yöntem büyük ağaçların oluşturulması sonucu elde edilen sapmanın düşük olması ve ağaçlar arasındaki düşük korelasyon nedeniyle daha iyi genellemeler sunar ve geçerli tahminlerde bulunabilir.

D. Cross Validation

Veri madenciliği çalışmalarında uygulanan "Cross Validation" literatürde k-fold cross validation yöntemi olarak geçer. Veri kümesi eğitim ve test kümeleri olarak ayrılır. %90 test, %10 eğitim(train) verilerinden oluşur. Bu ayırma işlemi çok sayıda farklı şekilde yapılabilir.

Cross validation'da öncelikle bir k değeri seçilir, veri kümesi k değeri kadar parçaya bölünür ve çapraz doğrulama sistemi çalışmaya başlar. Daha sonraki aşamada parçalar eğitim ve test verisi olarak ayrılır. K sayısı kadar parçalara bölünen veri setinde, sistemi çalıştırdıktan sonra bir sonuç elde ediliyor. Aynı işlem tüm parçalar seçilerek tekrarlanıyor. Sistemin genel anlamdaki başarısı ise elde edilen sonuçların ortalaması olarak hesaplanıyor.

Yapılan işlemlerin formülize edilmiş hali denklem 3 de gösterilmiştir.

$$t_i \in VK \text{ olmak üzere }, Sonuç = \frac{\sum_{i=0}^{k} SF(t_i, VK - t_i)}{k} \quad (3)$$

Yapılan işlemlerin formülize edilmiş hali

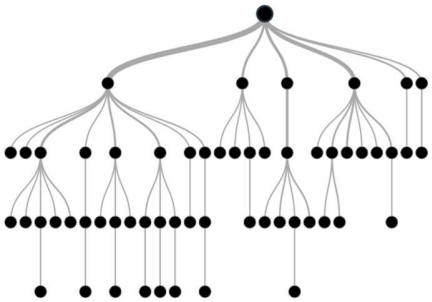
Formülde; SF (test, eğitim) sınıflandırma fonksiyonu, VK veri kümesi, k kaç parça katlama kullanıldığı ve t ise seçilen her bir test kümesi olarak verilmiştir. Sonuç tüm bu sınıflandırma fonksiyonlarının performanslarının toplamının k sayısına bölünerek ortalamasının alınmasına eşittir.

E. Desicion Tree

Karar ağacı, ağaç yapısı şeklinde bir sınıflandırıcı, gözetimli öğrenme algoritmasıdır. Çok sınıflı verilerin kullanıldığı veri setlerinde sınıflandırma ile sonuç çıkarımı için kullanılır ve çıkarımlar yapılmasını kolaylaştırır. Çok sayıda kayıt içeren veri kümesini daha küçük kümelere bölmek için kullanılır.Alt kümelerin oluşturulması, alt kümelerin homojenliğini artırır. Bölünmeler ne kadar iyi yapılmışsa, tahmin de o kadar iyidir. Karar ağacı öğrenmesi Sınıflandırma problemleri ve İlkelleme problemlerini çözmek amacıyla kullanılır. Karar ağacı algoritmasının kullanıldığı sınıflandırma problemlerinde kullanılacak veri seti eğitim(train) ve test olarak iki ana parçaya ayrılır. Algoritma, eğitim verilerini kullanarak model oluşturur. Oluşturulan bu model test verisi üzerinde uygulanarak modelin problem çözümündeki başarısı hesaplanır.Karar ağaçları mantıksal operatörler kullanılarak oluşturulur ve hem sayısal hem de kategorik değerler üzerinde işlemler yapar. Ağaç yapısı veri setine göre şekillenir. Yapısı itibariyle en tepeden en aşağı inen bir strateji sunmaktadır. Karar ağacı öğrenmesi sırasında, öğrenilen bilgi bir ağaç üzerinde modellenir. Bu ağacın bütün iç düğümleri (interior nodes) birer girdiyi ifade eder. Karar ağacı öğrenmesinde eğitim yapılan küme, alt kümelere bölünür ve bu işlem tekrarlanır. Tekrarlama işlemi, tahmin üzerinde bir etkisi kalmayana kadar sürer. Bu işleme özyineli parçalama(recursive partitioning) denir.

Karar ağacı öğrenme algoritmalarının anlaşılması ve yorumlanması basittir. Hızlı veri ön işlemesi gerektirir. Düşük hesaplama karmaşıklığına (computational complexity) sahip olmasından dolayı yüksek miktardaki veriyi kısa sürede işleyebilir. Karar ağacı algoritmaları beyaz kutu modelini kullanır, yani her adım görüntülenebilir ve yorumlanabilir. Bu algoritmaların bazı devavantajları vardır. İyileştirilmiş (optimum) bir karar ağacı öğrenme algoritması NP-Tam (NP-Complete) karmaşıklığa sahiptir. Bu yüzden bu algoritmada sezgisel algoritmalar (heuristic algorithms) veya aç gözlü

yaklaşımı (greedy approach) kullanılmaktadır. Bütün verinin özelliklerini modelleyemeyen bir ağaç ile karşılaşıldığında, budama(prunning) işlemi yapılmaktadır. ID3, C4.5, CHAID, CART en çok kullanılan karar ağacı algoritmalarındandır. **Şekil 4**'te karar ağacı yapısı görülmektedir. Ağaç çok fazla dallandığında budama işlemi de kararlılığı artırmak için uygulanılan tekniklerdendir.



Şekil. 4.Karar Ağacı yapısı

Karar ağacı yapısında, algoritma yapısı değişkenin tipine göre belirlenir. Kategorik değişkenler için Entropi, Gini; sayısal değişkenler için ise En Küçük Karalere yöntemi kullanılmaktadır. Entropi, verilerle ilgili belirsizliğin bir ölçüsüdür. Verilerin, entropiyi en aza indirecek şekilde bölünmesi gerekmektedir. Entropi Denklemi, denklem 4 de verilmiştir.

$$H = -\sum p(x) \log p(x) \tag{4}$$

p (x) belirli bir sınıfa ait grubun yüzdesini ve H ise entropiyi belirtmektedir.

Karar ağacı entropiyi en aza indirgeyen bölünmeler yapar. En iyi bölünmeyi belirlemek için de bilgi kazancı kullanılır. Bilgi kazancı hesaplaması denklem 5 da verilmiştir.

$$Gain(S, D) = H(S) - \sum_{V \in D} \frac{|V|}{|S|} H(V) \tag{5}$$

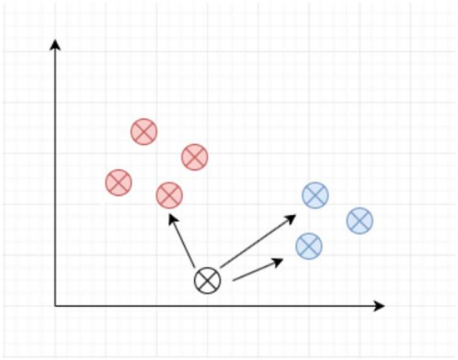
Bilgi Kazancı

S orijinal veri kümesidir ve D ise kümenin bölünmüş bir parçasıdır. Her V, S'nin bir alt kümesidir. V'nin tümü ayrıktır ve S'yi oluşturmaktadır. Bu durumda bilgi kazancı, bölünmeden önceki orjinal veri setinin entropisi ile her bir özniteliğin entropi değeri arasındaki fark olarak tanımlanmaktadır.

F. K-NN

KNN algoritması uygulaması kolay gözetimli öğrenme algoritmalarındandır. Çoğunlukla sınıflandırma problemlerinin çözümünde kullanılır. Algoritma, sınıfları belli olan bir örnek kümedeki verilerden yararlanılarak kullanılır. Örnek veri setine katılacak yeni verinin, mevcut verilere uzaklığı hesaplanır ve k sayıda yakın komşuluğuna bakılır. Bu hesaplama için genelde "Euclidean", "Manhattan" ve "Minkowski" uzaklık fonksiyonları kullanılır.

KNN; eski, basit ve gürültülü eğitim verilerine karşı dirençlidir. Fakat uzaklık hesabı yaparken bütün durumları sakladığından, büyük veriler için kullanıldığında çok sayıda bellek alanına gereksinim duymaktadır. **Şekil 5**'de örnek knn çizimi gösterilmiştir.



Şekil. 5.KNN gelen örneğin komşuluk gösterimi

KNN algoritmasında ilk olarak k parametresi belirlenir. K parametresi verilen bir noktaya en yakın komşuların sayısıdır. K sayıdaki komşuya göre sınıflandırma yapılmaktadır. Veri setine katılacak olan yeni verinin, mevcut verilere uzaklığı ilgili fonksiyonlar yardımıyla tek tek hesaplanır. En yakın k komşu dikkate alınır ve özellik değerlerine göre sınıflara atanır. Seçilen sınıf beklenen gözlem değerinin sınıfı olarak kabul edilir ve yeni veri label olur. Algoritmanın işleme adımları **Şekil 6** da görülmektedir



Şekil. 6.KNN algoritmasının adımları

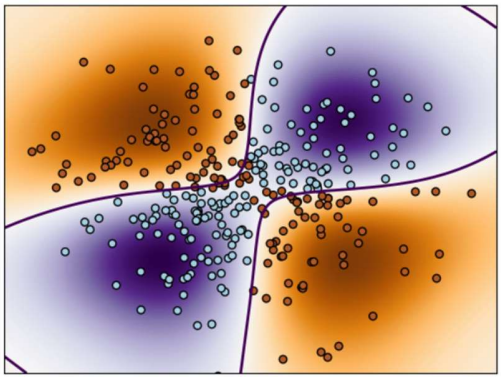
Eğitim kümesinin büyük olması ve k değerini uygun seçilmesi KNN açısından çok önemlidir. KNN algoritması uygulandığında, eğitim verisi ve k değeri arttırılarak başarı kontrol edilir. Başarının sabitlenmesi iyi bir KNN tahmin sistemi olduğunu belirtir. Algoritmada her ekleme işleminde uzaklıkların hesaplanır, bu yüzden yazılım süreci uzun sürebilir. Fakat veriye uygun filtreleme yöntemleri kullanılarak yazılım süreci hızlandırılabilir.

G. SVM

SVM (Destek Vektör Makinesi), iki sınıfa ait verileri birbirinden ayırmak için kullanılan çok yönlü ve çok güçlü bir makine öğrenmesi tekniğidir. İstatistiksel öğrenme teorisine ve yapısal risk minimizasyonuna dayanmaktadır. Sınıflandırma için düzlemde bulunan iki grup arasında bir sınır çizilerek iki grubu ayırmak mümkündür. Bu sınırın çizileceği yer ise iki grubun da üyelerine en uzak olan yer olmalıdır. SVM bu sınırın

nasıl çizileceğini belirler, bunun için karar sınırları / hiper düzlemler belirlenir. SVM her iki sınıfa da en uzak olan, sınıfları birbirinden ayıran ve özel bir çizgi olan hyperplane bulmayı amaçlar. SVM'ler nonparametric sınıflayıcılardır. Doğrusal ya da doğrusal olmayan sınıflandırma, regresyon, aykırı değer tespiti, yüz tanıma sistemleri ve ses analizine kadar birçok sınıflandırma probleminde kullanılmaktadır. Destek vektör makineleri boyut sayısının örneklem sayısından fazla olduğu yüksek boyutlu kümelerde etkilidir. Eğitim setlerinde girdi ve çıktılar eşlenir. Eşler aracılığıyla test verilerinde ve yeni veri setlerinde girdi değişkenini sınıflayacak karar fonksiyonları elde edilir. Karar fonksiyonunda support vektors denilen eğitim noktaları kullanılır. Böylece bellek verimli şekilde kullanılmış olur. DVM çok sayıda bağımsız değişkenle çalışmasına rağmen yüksek doğruluk oranına sahiptir ve karmaşık karar sınıflarını modelleyebilir.

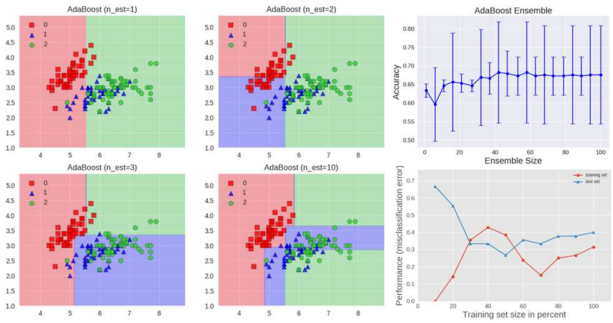
DVM, veri setinin özelliklerine göre, Doğrusal DVM, ve Doğrusal olmayan DVM şeklinde ikiye ayrılmaktadır. Linear DVM, verileri ayırabilecek sonsuz sayıda doğru içerisinden marjini en yüksek olan doğrunun seçilmesidir. NonLinear DVM'de ise mapping ile çalışma verisi yüksek boyuta dönüştürülür ve verinin taşındığı yeni boyutta marjini en uygun ayırıcı araştırma işlemidir. Algoritmanın oluşturduğu sınıflar **Şekil 7'**de görülmektedir.



Şekil. 7.Doğrusal olmayan DVM

H. Adaboosting

Adaboost algoritmasında her temel öğrenici, derinliği 1 olan bir karar ağacından oluşur, böylece verileri, eksenlerden birine paralel olan doğrusal bir karar yüzeyi ile ayrılan alanı iki bölgeye ayıran bir özellik eşiğine göre sınıflandırır. Algoritmanın öğrenme aşamaları **Şekil 8**'te gösterilmiştir. Şekilde ayrıca, testin doğruluğunun grubun büyüklüğü ve eğitim ve test verileri için öğrenme eğrileri ile nasıl geliştiği gösterilmektedir.



Şekil. 8.Adaboosting algoirtması öğrenme aşamaları

İ. Bootstrapping

Bootstrapping tekniği, rastgele örnekleme yöntemlerini kullanarak tüm istatistiklerin örnekleme dağılımını tahmin etmeyi sağlar. Yeniden örnekleme yöntemlerinin daha geniş bir sınıfına ait olduğu söylenebilir. Bootstrapping, (varyans gibi) bir aracın özelliklerini hesaplarken tahmini bir dağılımdan yararlanma eylemidir. Tahmini dağılım için standart seçimlerden birisi gözlemlenen verinin yapay dağılım fonksiyonudur. Eğer bu gözlemler bağımsız ve birebir dağılmış popülasyonlara sahipse bu seçenek yerleştirmeyle bir çok yeni örneklem(resamples) oluşturularak kullanılır. Bootstrapping ayrıca hipotez testleri oluşturmak için de kullanılabilir.

J. Bagging

Bagging, bootstrap birleşmelerinden oluşur. Bir tahminin varyansını azaltmanın yollarından biri de birden fazla hesaplamanın ortalamasını almaktır. Örneğin, yeniden yerleştirmeyle seçilmiş farklı data alt kümelerinin üzerinde M tane farklı ağaç eğitilip ensemble hesaplanabilir.

$$f(x) = 1/M \sum_{m=1}^{M} f_m(x) \quad (6)$$

Bagging, temel öğrenenleri eğitmek ve veri altkümelerini elde etmek için bootstrapping örneklemesini kullanır. Bagging temel öğrenenlerin çıktılarını toplamak, sınıflandırmak ve regresyon için ortalamayı kullanır.

V. VERİ SETİ

Kitsune 2018 yılında "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection" ismi ile yayınlanmış olan normal paketlerden 115 tane özellik üreterek, denetimsiz olarak saldırıları tespit eden açık kaynak NIDS tasarlanmıştır. Yapısı **Şekil 9**'de göründüğü gibi tasarlanmıştır. Kitsune FE ve FM bizim projemizde de aynen kullanılmıştır. Üretilen veri seti ön işleme yapılarak saldırı etiketleri veri setinin diğer özellikleri ile birleştirilmiştir.

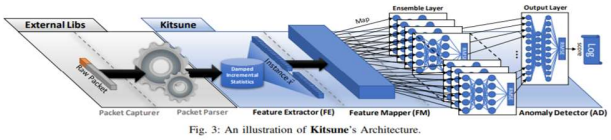
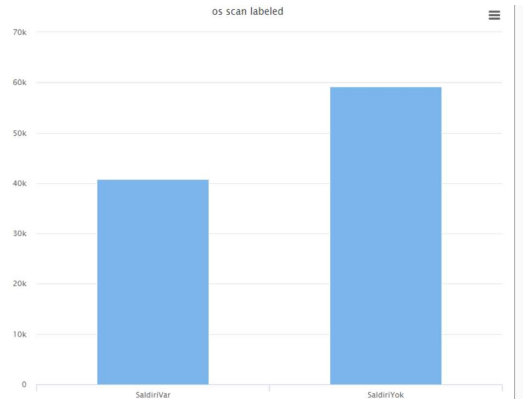


Fig. 3: An illustration of **Kitsune**'s Architecture.

Şekil. 9. Kitsune saldırı tespit sistemi yapısı

Veri setinden bootstrapping yöntemi ile 100.000'lik bir örneklem alınarak ensemble yöntemler kullanılarak uygulanmıştır. Veri setimiz **Şekil 10**'de görüldüğü üzere yaklaşık %60'ı normal trafik, %40'ı OS Scan saldırısından oluşmaktadır.



Şekil. 10. Veri seti saldırı ve normal ağ trafiğinin dağılımı

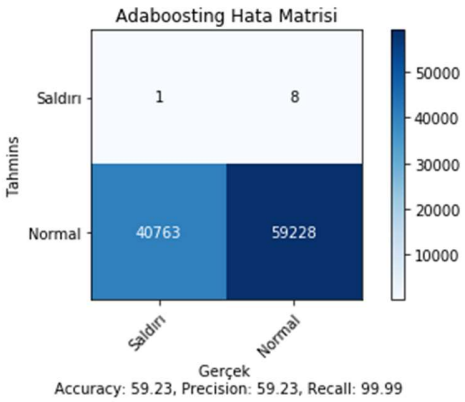
VI. BULGULAR

Hackerlar saldıracakları ağlara saldırmadan önce keşif saldırıları yaparlar. Bu keşif saldırılarından en önemlileri ağ bağlı cihazlarda port ve işletim sistemi taraması yapılması adımıdır. Çünkü açık olan portlara ve kullanılan işletim sistemine göre yapılacak saldırılar çeşitlilik göstermektedir. [4]

Os scan yapılırken ağdaki paket trafik hareketlerini makine öğrenmesi algoritmaları ile normal trafikten ayırt etmek mümkündür [1]. Bu saldırı trafiğini deneysel olarak hazırlanmış Kitsune veri seti üzerinde değerlendirilip bir saldırı tespit modeli oluşturulmaktadır. Yapılan çalışmada Kitsune veri seti içerisindeki Os scan saldırısı için denetimli öğrenme işlemi gerçekleştirilmiştir. Bu makine öğrenmesi yöntemlerinden Adaboosting, Decision Tree, K-NN, Navie Bayes, Random Forest ve Support Vektör Machine (SVM) algoritmaları kullanılarak saldırı tespit sistemi tasarlanmıştır. Bootstraping örneklem alma yöntemi ile Kitsune os scan veri setinden 100.000 örnek kullanılarak algoritmalardan sonuçlar alınmıştır.

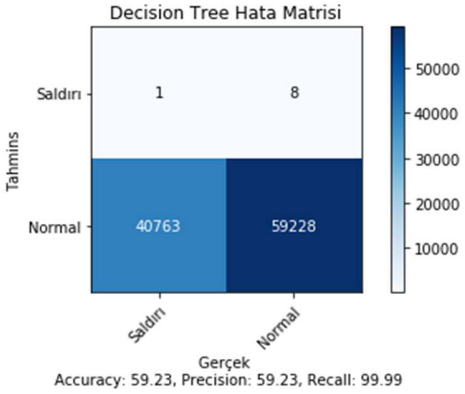
Adaboosting algoritması sınıflandırma için kullanıldığında hata matrisi Tablo 1'de verilmiştir. Doğruluk (Accuracy) %59.23 ve Hassaslık (Precision) %59.23 sonuçlarını verirken algoritmalar arasında en iyi Recall (Duyarlılık) değeri %99.99 olarak hesaplanmıştır. Adaboosting öğrenme ve test süresi 100.000 örnek için 16 GB ram 8. Nesil İntel i7 kişisel bilgisayarda 4 dakika 28 saniye sürmüştür.

TABLE I. 100.000 ÖRNEK İÇİN EĞİTİLEN ADABOOSTİNG HATA MATRİSİ



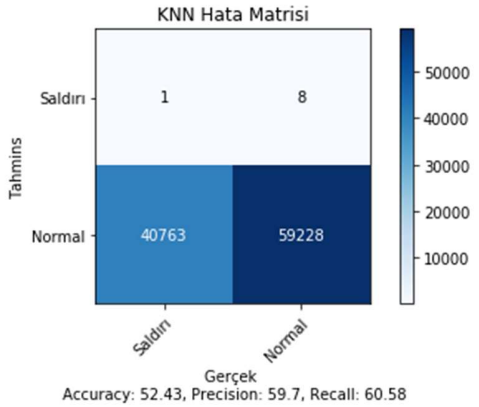
Decision Tree algoritması sınıflandırma için kullanıldığında hata matrisi Tablo 2'de verilmiştir. Doğruluk (Accuracy) %59.23, Hassaslık (Precision) %59.23 sonuçlarını verirken algoritmalar arası en iyi Recall değeri %99.99 olarak hesaplanmıştır. Decision Tree öğrenme ve test süresi 16 GB ram 8. Nesil İntel i7 kişisel bilgisayarda 1 dakika 13 saniye sürmüştür.

TABLE II. 100.000 ÖRNEK İÇİN EĞİTİLEN DECİSİON TREE HATA MATRİSİ



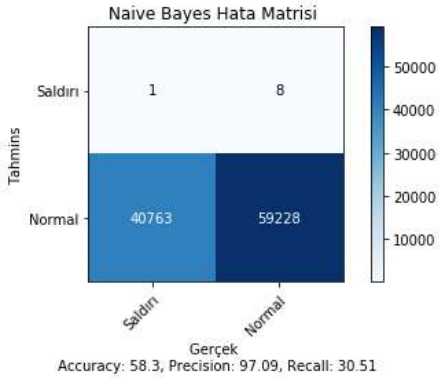
KNN algoritması sınıflandırma için kullanıldığında hata matrisi Tablo 3'te verilmiştir. Algoritmalar arasındaki en düşük accuracy değerine sahiptir. Doğruluk (Accuracy) %52.43, Hassaslık (Precision) %59.7 ve Recall (Duyarlılık) %60.58 olarak hesaplanmıştır. KNN öğrenme ve test süresi 16 GB ram 8. Nesil İntel i7 kişisel bilgisayarda 19 dakika 19 saniye sürmüştür.

TABLE III. 100.000 ÖRNEK İÇİN EĞİTİLEN KNN HATA MATRİSİ



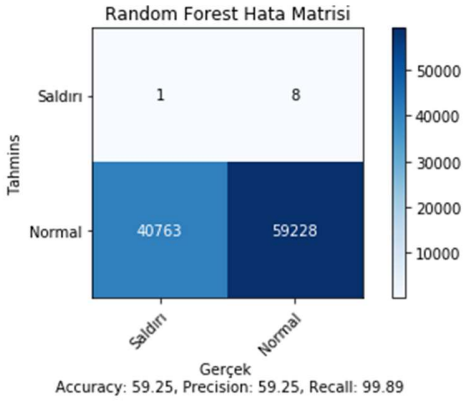
Navie Bayes algoritması sınıflandırma için kullanıldığında hata matrisi Tablo 4'te verilmiştir. Doğruluk değeri (Accuracy) %58.3 olarak hesaplanmıştır. %97.09 ile algoritmalar arasında en yüksek Hassaslık (Precision) değerine ve %30.51 ile en düşük Recall (Duyarlılık) değeri olarak hesaplanmıştır. Navie Bayes öğrenme ve test süresi 16 GB ram 8. Nesil İntel i7 kişisel bilgisayarda 2 saniye sürmüştür.

TABLE IV. 100.000 ÖRNEK İÇİN EĞİTİLEN NAVİE BAYES HATA MATRİSİ



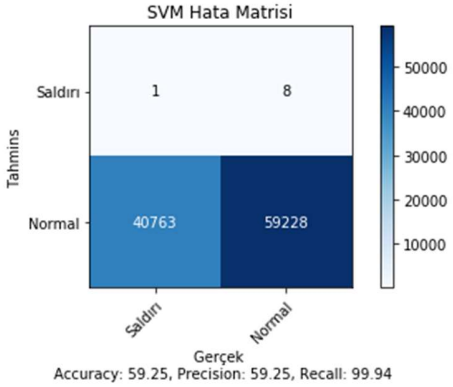
Random Forest algoritması sınıflandırma için kullanıldığında hata matrisi Tablo 5'te verilmiştir. Algoritmalar arasında %59.25 ile en yüksek doğruluk (Accuracy) değerine sahiptir. Hassaslık (Precision) değeri %59.25 ve Recall (Duyarlılık) değeri ise %99.89 olarak hesaplanmıştır. Random Forest öğrenme ve test süresi 16 GB ram 8. Nesil İntel i7 kişisel bilgisayarda 2 saniye sürmüştür.

TABLE V. 100.000 ÖRNEK İÇİN EĞİTİLEN RANDOM FOREST HATA MATRİSİ



SVM algoritması sınıflandırma için kullanıldığında hata matrisi Tablo 6'da verilmiştir. %59.25 değeri ile algoritmalar arasındaki en yüksek doğruluk (Accuracy) değerine sahiptir. Hassaslık (Precision) %59.25 ve Recall (Duyarlılık) %99.94 olarak hesaplanmıştır. SVM öğrenme ve test süresi 100.000 örnek için 16 GB ram 8. Nesil İntel i7 kişisel bilgisayarda 4 saat 20 dakika 3 saniye sürmüştür.

TABLE VI. 100.000 ÖRNEK İÇİN EĞİTİLEN SVM HATA MATRİSİ



K-NN algoritması % 52.43 doğruluk (accuracy) oranı ile algoritmalar arasında en düşük doğruluk oranına sahip algoritma olmuştur.

Kitsune veri setinde OS Scan saldırısının tamamı cross validation kullanılarak Adaboosting, Decision Tree, KNN, Navie Bayes, Random Forest ve SVM algoritmaları uygulandığında Accuracy, Precision ve Recall (Duyarlılık) değerleri Tablo 7 de verilmiştir. Algoritmalar kıyaslandığında accuracy %59.25 ile en yüksek değerlere Random Forest ve SVM algoritmasında ulaşılmıştır. Precision değerinde %97.09 ile Navie Bayes, Recall değerinde ise %99.99 ile Decision Tree ve Adaboosting algoritmalarında en yüksek değerler elde edilmiştir.

TABLE VII. KULLANILAN ALGORİTMALARIN DEĞERLERİNİN KARŞILAŞTIRMASI

Algorithm	Accuracy (Doğruluk) (%)	Precision (Hassasiyet) (%)	Recall (Duyarlılık) (%)	Working Time (Çalışma Zamanı)
ADABOOSTİNG	%59.23	%59.23	%99.99	4:28
DECİSİON TREE	%59.23	%59.23	%99.99	1:13
K-NN	%52.43	%59.70	%60.58	19:19
NAVİE BAYES	%58.30	%97.09	%30.51	00:02
RANDOM FOREST	%59.25	%59.25	%99.89	00:02
SVM	%59.25	%59.25	%99.94	4:20:03

VII. SONUÇ VE TARTIŞMA

Bu çalışmada bootstrapping örneklem alma yöntemi ile Kitsune os scan veri setinden 100.000 örnek kullanılarak algoritmalardan sonuçlar alınmıştır. Os scan saldırısını tespit etmek için bir modul ortaya konulmuştur. Cross validation kullanılarak SVM, KNN, Navie Bayes, Random Forest, Decision Tree ve Adaboosting algoritmaları karşılaştırılmıştır. Yapılan bu çalışmada os scan saldırısının network üzerinde makine öğrenmesi yöntemleriyle tespit edilebileceği gösterilmiştir. Deneyler sonucunda Kitsune os scan veri setinde Random Forest ve SVM %59.25 ile en yüksek doğruluk değerine sahip algoritmalar olmuştur. Bu algoritmaların doğruluk değerine en uzak değere sahip algoritma ise KNN'dir. Çalışma süresi olarak değerlendirildiğinde Navie Bayes ve Random Forest en hızlı sonuç veren algoritmalar olurken, SVM ise en yavaş sonuç veren algoritma olmuştur.

Sığ yöntemler olarak bilinen klasik yöntemlerle bu saldırıyı tespit etmeye çalışılmıştır. Alınan sonuçlar os scan tespiti için kullanılan paketlerin networkteki diğer paketlere benzemesi dolayısıyla tespit edilmesi zor bir saldırı olarak karşımıza çıkmaktadır. Bu sonuçlar yapılan çalışma için yeterli bulunmamış ancak ara sonuçlar olarak literatürde paylaşılması uygun görülmüştür. Sonuçları iyileştirmek için farklı modeller denenmesi ve os scan keşif saldırısı tespiti için model geliştirilmesi planlanmaktadır. Kullanmış olduğumuz yöntemlere ilaveten Neural Network ve Deep Learning algoritmaları ile de yeterli doğruluk oranlarına ulaşılamamıştır. Kullanmış olduğumuz veri seti için Neural Network %58.92, Deep Learning %59.24 doğruluk oranı vermiştir.

Gelecek çalışmalarda aldığımız sonucun doğruluğunu artırmak için PCA analiz, feature extraction gibi yöntemler kullanılması planlanmaktadır. Algoritmaların doğruluğu artırıp artırmadığını öğrenmek ve sonuçları iyileştirmek için, veri setinden daha büyük bir örneklem veya veri setinin tamamı ile daha güçlü bir veri işleme ortamında bu çalışmanın tekrarlanması planlanmaktadır.

BİLGİLENDİRME

Bu çalışma İstanbul Kalkınma Ajansı tarafından desteklenen Nesnelerin İnterneti Güvenlik Test ve Değerlendirme Merkezi araştırmacıları tarafından hazırlanmıştır.

REFERENCES

[1] Y. Mirsky, T. Doitshman, Y. Elovici, A. Shabtain, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," Network and Distributed Systems Security Symposium (NDSS), 2018.

[2] S. C. Lee and D. V. Heinbuch, Training a Neural-Network Based Intrusion Detector To Recognize Novel Attacks, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 2001, pp. 294-299.

[3] A. N. Singh and R.C. Joshi, A Honeypot System For Efficient Capture And Analysis Of Network Attack Traffic, 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies, 2011.

[4] G. Conti and K. Abdullah, Passive Visual Fingerprinting of Network Attack Tools, ACM Conference on Computer and Communications Security's Workshop on Visualization and Data Mining for Computer Security (VizSEC), 2004.

[5] T. H. Ptacek and T. N. Newsham, Insertion, evasion, and denial of service: Eluding network intrusion detection, Secure Networks, Inc., Calgary, AB, Canada, 1998.

[6] C. B. Lee, C. Roedel and E. Silenok Detection and Characterization of Port Scan Attacks, Department of Computer Science & Engineering University of California, San Diego, 2003.

[7] J. Gadge and A. A. Patil, Port Scan Detection, 2008 16th IEEE International Conference on Networks, New Delhi, 2009.

[8] A. L. Buczak and E. Guven, A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection, IEEE Communications Surveys & Tutorials (Volume: 18 , Issue: 2 , Secondquarter 2016), 2015, pp. 1153-1176.

[9] K. Ghanem, Francisco J. Aparicio-Navarro, K. G. Kyriakopoulos, S. Lambotharan and J. A. Chambers, Support Vector Machine for Network Intrusion and Cyber-Attack Detection, 2017 Sensor Signal Processing for Defence Conference (SSPD), London, 2017.

[10] S. Sahu and B M Mehtre, Network intrusion detection system using J48 Decision Tree, 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, 2015.

[11] M. Kumar, M. Hanumanthappa and T. V. Suresh Kumar, Intrusion Detection System using decision tree algorithm, 2012 IEEE 14th International Conference on Communication Technology, Chengdu, 2012.

[12] S. M. Almansob, S. S. Lomte, Addressing challenges for intrusion detection system using naive Bayes and PCA algorithm, 2017 2nd International Conference for Convergence in Technology (I2CT), Mumbai, 2017.

[13] Weiming Hu, Wei Hu and S. Maybank, AdaBoost-Based Algorithm for Network Intrusion Detection, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) (Volume: 38 , Issue: 2 , April 2008), 2008, pp. 577-583.

[14] L. Breiman, Random Forests, University of California, Berkeley, 2001.

[15] J. Zhang and M. Zulkernine, Network Intrusion Detection using Random Forests, Queens University, Kingston, Ontario, 2005.

[16] F. Chen, Z. Ye, C. Wang, L. Yan and R. Wang, A Feature Selection Approach for Network Intrusion Detection Based on Tree-Seed Algorithm and K-Nearest Neighbor, 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Lviv, 2018

[17] G. Xiaobing, Q. Depei, L. Min, Z. Ran and X. Bin, Detection and protection against network scanning: IEDP, Proceedings 2001 International Conference on Computer Networks and Mobile Computing, Beijing, 2001.

[18] R. Ensafi, J. C. Park, D. Kapur, J. R. Crandall, Idle Port Scanning and Non-interference Analysis of Network Protocol Stacks Using Model Checking. 19th USENIX Security Symposium, Washington, 2010.

[19] A. akn and M. A. Aydn, "IPv4 / IPv6 Gvenlik Tehditleri ve Karlatrlmas," Department of Informatics, Yldz Technical University, Department of Computer Engineering, Istanbul University, Istanbul.

[20] Agenda and Work Plan. Computer Security Incident Response Team (CSIRT), Florida State University