YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GERÇEK ZAMANLI VERİLER YARDIMI İLE KARAR VEREN BİR BİLGİSAYAR AĞI SALDIRI TESPİT SİSTEMİNİN TASARLANMASI VE GERÇEKLENMESİ

Bilgisayar Mühendisi Erdem CAN

FBE Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı: Yrd. Doç. Dr. A. Gökhan YAVUZ (YTÜ)

İSTANBUL, 2007

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ

ARP	Address Resolution Protocol
CERT/CC	Coordination Centre of the Computer Emergency Response Team at Carnegie-Mellon University
CIDF	Common Intrusion Detection Framework
DARPA	Defense Advanced Research Project Agency
DIDS	Distributed Intrusion Detection System
DNS	Domain Name System
DPEM	Distributed Program Execution Monitoring
EMERALD	Event Monitoring Enabling Responses to Anomalous Live Disturbances
FP	Frame Pointer
FTP	File Transfer Protocol
HMM	Hidden Markov Model
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDES	Intrusion Detection Expert System
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IP	Internet Protocol
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
KDD	Knowledge Discovery and Data Mining
LAN	Local Area Network
MTU	Maximum Transmission Unit
NIDES	Next Generation Intrusion Detection Expert System
NIDS	Network Intrusion Detection System
NIST	National Institute of Standards and Technology
NSM	Network Security Monitor
OSI	Open Systems Interconnection
PC	Personal Computer
PGP	Pretty Good Privacy
SEI	Software Engineering Institute
SNMP	Simple Network Management Protocol
SP	Stack Pointer
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Socket Layer
STAT	State Transition Analysis Tool
SVM	Support Vector Machines
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WAN	Wide Area Network
WWW	World Wide Web

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖNSÖZ

Hayatımın her anında yanımda olan ve her konuda desteklerini esirgemeyen babama, anneme, ablama ve kardeşime; her gün üniversiteye mutlu bir şekilde gelme sebebim olan tüm çalışma arkadaşlarıma ve değerli hocalarıma; gerek lisans gerekse de yüksek lisans eğitimimde ve tez geliştirme sürecinde büyük bir sabır ile içten yardımlarını, engin bilgisini ve değerli zamanını esirgemeyen danışman hocam Yrd. Doç. Dr. A. Gökhan YAVUZ'A teşekkürlerimi sunarım.

ÖZET

Saldırı tespit sistemleri anormallik tespiti ve imza tabanlı olmak üzere iki farklı yaklaşım içerir. Bu yaklaşımların birbirlerine göre avantaj ve dezavantajları vardır. Tez çalışmasında her iki yaklaşımı da kullanarak saldırı tespit başarısını arttırmak amaçlanmıştır. Anormallik tespiti için SVM (Destek Vektör Makinesi) metodu kullanılmıştır. Ancak, bazı sınıflandırma algoritmalarının da tasarlanan sistem üzerindeki başarımları ölçülmüştür İmza tabanlı yaklaşım için Snort [3], (Rehman, 2003) sisteminden yararlanılmıştır. Snort sisteminin yalın ve modüler yapıda oluşu; imza veritabanının sürekli güncellenmesi; gerçek zamanlı ve anormallik tabanlı çalışan bir sistemin Snort yapısının üzerine inşa edilmesine olanak tanımaktadır. Tez çalışmasının gerçeklenmesinde hem açık kaynak oluşu hem de uygulama geliştirme ortamlarının zenginliği nedeni ile Linux işletim sistemi tercih edilmiştir. Tez çalışmasının sonucu diğer saldırı tespit sistemleriyle karşılaştırılmış ve yapılan çalışmanın başarılı sonuçlar ürettiği görülmüştür.

Anahtar Kelimeler: Saldırı, saldırı tespiti, ağ saldırı tespit sistemi, Snort, destek vektör makinesi, C4.5, Random Forest, DARPA veri kümesi, KDD Cup 99 veri kümesi

ABSTRACT

Intrusion detection systems consist of two different approaches, anomaly detection and signature based detection. Both approaches have advantages and disadvantages against the other. In this thesis, it is aimed to use both approaches to increase the success rate of the intrusion detection system. It is considered to use the Snort system [3], (Rehman, 2003) for the signature based approach. Constantly updating the signatures used in the Snort system and the basic and modular structure of the system allows a real-time and anomaly based system to be built on a Snort structure. To detect anomalies it is considered to use SVM (Support Vector Machines) method. Linux Operating System is used to implement the application because of its open-source structure and the richness of its application developing environment.

Keywords: Intrusion, Intrusion detection, Intrusion detection system, Snort, Support vector machine, C4.5, Random Forest, DARPA intrusion detection evaluation data, KDD Cup 99 data

1. GİRİŞ

Son 20 yılda bilgi teknolojilerindeki hızlı gelişmeye paralel olarak bilgi güvenliği konusu giderek önem kazanmıştır. Bilgi güvenliği kavramının oluştuğu ilk zamanlarda sunucu sistemlerindeki veri ve erişim güvenliği konusunda yapılan çalışmalar, Internet'in icadı ve hızla yaygınlaşması ile bilgisayar ağları üzerinde iletilen verinin güvenliğini de kapsamıştır. Bugün bilgi iletiminin büyük bir bölümü Internet ve kurum içi bilgisayar ağları üzerinden yapılmaktadır. Bilgisayar ağları, bilgisayarımızın dış dünyaya açılan kapısıdır. Bilgisayar ağı üzerinden dış dünyaya bağlanan bilgisayarlar, aynı zamanda dış dünyadaki kötü niyetli kullanıcılar tarafından gerçekleştirilecek saldırılara açık hale gelirler. Bilgisayar ağı üzerindeki bilgisayarların ve bilginin güvenliğini sağlamak için şifreleme, doğrulama, erişim kontrolü, bütünlük kontrolü, sistem günlüklerinin kullanımı ve güvenlik duvarı gibi doğrudan güvenlik sağlayan yöntemler mevcuttur. Ancak, bu yöntemler güvenliğin sağlanması için her zaman yeterli olmayabilir veya uygulanmalarında eksiklikler olabilir. Saldırganlar kendilerini bu yöntemlere göre geliştirebilir ve bu güvenlik önlemlerini aşabilirler. Bilgisayar sistemimizi veya bilgisayar ağımızı zayıf bir şekilde yakalayabilirler. Yapılan araştırmalara göre saldırıların %80'i kurum içi geri kalan %20'si ise kurum dışı kaynaklardan gelmektedir. Güvenlik önlemleri genellikle dış tehditler göz önüne alınarak düzenlenir. Ancak, kurum içi kaynaklara sınırlı erişim hakkı olan kişilerin bu yetkilerini kötüye kullanmaları veya sistemlere kasıtsız da olsa zarar verme riskleri daha yüksektir. Böyle bir durumda oluşan tehdidi algılayacak ve bize haber verecek bir sistemin varlığı büyük önem taşımaktadır. Bilgisayar ağı üzerinde veya bilgisayar sistemindeki veriyi ve her türlü kaynağı iç veya dış tehditlere karşı denetleyen sistemlere saldırı tespit sistemleri denir.

Saldırı tespit sistemlerinin tarihçesi 1980'lere kadar uzanır. İlk saldırı tespit sistemleri çok kullanıcılı bilgisayar sistemlerindeki kaynak ve erişim güvenliğini sağlamak üzere tasarlanmışlardır. Bu sistemlerin başlıcası 1984 yılında Dorothy Denning ve Peter Neumann tarafından geliştirilmeye başlanan IDES'tir (Saldırı Tespit Uzman Sistemi). Dorothy Denning'in raporunda normal aktivitenin bir modeli oluşturularak normal dışı aktivitelerin ortaya çıkarılabileceği öne sürülmüştür (Denning, 1987). 1980'lerin sonunda istatistiksel ve uzman sistemlerin kullanıldığı HayStack (Smaha et al., 1988) ve NADIR (Hochberg et al., 1993) uygulamaları ortaya atılmıştır. Bunların dışında başlıca sistemler durum geçiş analizini kullanan USTAT (Ilgun, 1992) ve EMERALD'dır (P.A.Porras & P.G.Neumann, 1997).

Günümüzde ise gerek açık kaynak kodlu olması gerekse de basit ve hızlı karar verebilen kurallardan oluşması sebebi ile Snort [3], (Rehman, 2003) yaygın olarak kullanılmaktadır.

Tez çalışmasında gerçek zamanlı bir bilgisayar ağı saldırı tespit sistemi tasarlanmış ve uygulanmıştır. Sistemde hem imza tabanlı hem de anormallik temelli saldırı tespiti yaklaşımları kullanılmıştır. İmza temelli yaklaşım için Snort yazılımından yararlanılmıştır. Sistem, Snort yazılımına ön işlemci eklentisi olaracak şekilde geliştirilmiştir. Böylece Snort yazılımına anormallik tespiti özelliği kazandırılmıştır. Sistem, gerçek zamanlı çalışacak şekilde yalın tasarlanmıştır. Tasarlanan sistem farklı öğrenme algoritmaları ve saldırı tespit sistemleri ile karşılaştırılarak sistemin başarısı ölçülmüştür.

Tez kitabının ikinci bölümünde temel kavramlar üzerinde durulmuştur. Saldırı tespiti ve saldırı tespit yaklaşımları, temel saldırı tespit sistemleri, farklı ağ katmanları ve protokollerinde yapılan başlıca yetkisiz faaliyetler anlatılmıştır. Üçüncü bölümde sistemin tasarımı verilmiştir. Dördüncü ve beşinci bölümlerde tasarımda kullanılan fonksiyon ve veri yapıları detaylı olarak incelenmiştir. Altıncı bölümde sistemde yapılan iyileştirmeler, karşılaştırmalar ve sınamalar anlatılmıştır. Sonuç bölümde ise çalışma ile ilgili değerlendirmelere yer verilmiştir.

2. TEMEL KAVRAMLAR

Bu bölümde bilgi ve bilgi güvenliği, saldırı tespiti, saldırı tespit sistemleri ve yetkisiz faaliyetler konularında temel kavramlar ele alınmıştır.

2.1 Bilgi ve Bilgi Güvenliği

Bilgi, kurallardan yararlanarak kişinin veriye yönelttiği anlamdır [1]. Bilmek ve bilineni muhafaza etmek insanları diğer canlılardan üstün kılmaktadır. Bilgi sayesinde insanlık sürekli öğrenmiş ve öğrendiklerini diğer nesillere aktarabilmiştir. Bilgiyi korumak ve güvenliğini sağlamak insanlar için çok büyük önem taşımaktadır. Günümüzde bilginin saklanması ve taşınması fiziksel ortamdan sayısal ortama kaymıştır. Sayısal bilgi güvenliği iki ana başlıkta incelenebilir. Bilginin güvenli bir şekilde sayısal ortamda saklanmasına *bilgisayar güvenliği*, bilginin bilgisayar - bilgisayar veya insan - bilgisayar arasında güvenli olarak iletimini sağlamak üzere oluşturulan mekanizmalara *bilgisayar ağ güvenliği* adı verilir. Bilgi güvenliği konusuna değinmeden önce birkaç temel kavramı açıklamak gerekmektedir.

Zayıflık (Vulnerability): Bir bilişim sisteminin tehditlerden etkilenmesine neden olabilecek açıkları.

Tehdit (Threat): Sistemin zayıflıklarını kullanarak sisteme zarar verici davranışlarda bulunma olasılığı olan eylemler. Tehdit, zayıflıkları istismar edebilecek potansiyel tehlikedir. Tehdit bilinçli (örn. güvenlik kırıcılar, suç organizasyonları) veya bilinçsiz (örn. bilgisayar arızaları veya doğal afetler) olabilmektedir.

Bilgisayar güvenliği konusundaki ilk kapsamlı çalışmalardan birini yapan James P. Anderson , Çizelge 2.1'de gösterildiği üzere tehditleri geldikleri yere göre 3 ana gruba ayırmıştır.

Çizelge 2.1 Tehdit Sınıfları (Anderson, 1980).

		Kaynağa Erişim	
		Yetkisiz	Yetkili
Sisteme Erişim	Yetkisiz	**A** Dış Kaynaklı Tehdit	
	Yetkili	**B** İç Kaynaklı Tehdit	**C** Kötüye Kullanım

A tehdit grubundakiler sisteme ve kaynaklara erişimleri olmadıklarından dış kaynaklı tehdit unsuru olarak adlandırılırlar. **B** tehdit grubundakiler sisteme kısıtlı erişme hakları varken yetkisiz oldukları kaynaklara izinsiz erişmek istemektedirler. **C** tehdit grubundakiler ise sistem ve kaynağa erişim yetkileri varken bu kaynakları yetkisiz kişilere vererek yetkilerini kötüye kullananlar olarak tanımlanabilir.

Saldırı (Attack): Kurum ve şahısların sahip oldukları bilgilere yetkisiz erişmek, zarar vermek, maddi/manevi kazanç sağlamak vb. için bilişim sistemleri kullanılarak yapılan her türlü hareket.

Saldırgan (Attacker): Kasıtlı bir şekilde, sistemin zayıflıklarından faydalanarak, sistem üzerindeki hizmetlere, veriye veya işleyişe tehdit olan hareket veya durumları oluşturan kişidir.

İçeri Sızma (Intrusion): Bilgisayar sistemine, güvenlik önlemlerini aşarak yetkisi olmadan erişme.

İzinsiz Kullanıcı: Bilgisayar sistemlerine yetkisi olmadan giriş yapan kişidir.

Bilgi güvenliğini sağlamak için organizasyonların öncelikle güvenlik gereksinimlerini belirlemeleri ve bu gereksinimleri karşılayacak doğru yaklaşımları tanımlamaları beklenmektedir. Bu birçok organizasyon için çok zor bir işlemdir. ITU-T (International Telecommunication Union – Telecommunication Standardization Sector), bilgi güvenliğini sağlamada yön göstermesi amacıyla, OSI (Open Systems Interconnection) güvenlik

mimarisini oluşturmuştur. OSI güvenlik mimarisi güvenlik konusunu üç ana başlık altında toplar (Stallings, 2006).

- Güvenlik Atağı
- Güvenlik Servisi
- Güvenlik Mekanizması

2.1.1 Güvenlik Atakları

Organizasyona ait olan bilginin güvenliğini tehlikeye atan her türlü eylemdir. Güvenlik atakları pasif ve aktif ataklar olmak üzere iki alt kategoride incelenebilir.

2.1.1.1 Pasif Ataklar

Sistem hakkında bilgi edinmeye çalışan ancak sistemin işleyişine veya kaynaklarına etki etmeyen ataklardır. Pasif atakların doğasında iletişimi gizlice dinleme ve izleme yer alır. Saldırganın hedefi iletişim halindeki bilgiyi ele geçirmektir. İki çeşit pasif atak vardır: **mesaj içeriğini okuma** ve **trafik analizi**. Bir telefon görüşmesi, bir elektronik posta mesajı veya transfer edilen bir dosya hassas ve gizli bilgi içerebilir. Ancak; bu tür iletişim yöntemleri veriyi düz metin olarak transfer ettiklerinden mesaj içeriğini okumak mümkündür. Trafik analizinde ise transfer edilen bilgi şifreleme yöntemleri kullanılarak gizlenir. Transfer edilen bilgiye pasif atak yapan saldırganın bilgiyi öğrenme ihtimali çok zordur. Ancak şifrelenmiş mesajdaki örüntüleri araştırarak iletişimin kimler tarafından yapıldığını, iletişimin yoğunluğunu ve mesajların boyutu gibi bilgileri elde edebilir. Bu gibi bilgiler saldırgana bir sonraki adım için yol gösterici olabilir. Pasif atakları yakalamak son derece zordur çünkü veri üzerinde hiçbir değişiklik yapmamaktadırlar. Haberleşme normal olarak devam eder ve ne gönderici ne de alıcı üçüncü bir kişinin mesajları okuduğundan veya trafiği analiz ettiğini fark etmez. Bu tür atakların başarısız kılınabilmesi için yapılabilecek yöntemler şifreleme metotlarını ve gelişmiş anahtarlama cihazlarını kullanmaktır. Pasif atakları tespit etmekten ziyade korunma amaçlı önlemler almak üzerinde yoğunlaşmak gerekmektedir.

2.1.1.2 Aktif Ataklar

Aktif ataklar veri akışını değiştirir veya yanlış veri akışı oluştururlar. Aktif ataklar, pasif ataklara göre tam tersi bir karakteristik sergilemektedirler. Pasif atakları her ne kadar belirlemek zor da olsa bu tür saldırıların başarısını engelleyen tedbirler mevcuttur. Diğer taraftan, aktif atakları tamamıyla engellemek de çok güçtür. Çünkü tam güvenlik için tüm iletişim ortamlarına ve yollarına her zaman fiziksel koruma sağlanması gerekmektedir. Bunun yerine hedef, saldırıları tespit etmek ve sistemdeki herhangi bir bozulma veya gecikmeyi geri döndürebilmektir. Saldırı tespitinin caydırıcılığı saldırıyı önlemede katkı sağlamaktadır. Aktif ataklar beş alt gruba ayrılır (Schneier, 1993).

- Maskeleme
- Tekrarlama
- Değiştirme
- Üretme
- Engelleme

2.1.2 Güvenlik Servisleri

Organizasyona ait veri işleme sistemlerindeki veri transferlerinin güvenliğini arttıran hizmetlerdir.

- Kimlik Doğrulama (Authentication)
- Erişim Kontrolü (Access Control)
- Veri Gizliliği (Data Confidentiality)
- Veri Bütünlüğü (Data Integrity)
- İnkar Edilemezlik (Non-repudiation)
- Erişilebilirlik (Accessibility Service)

2.1.3 Güvenlik Mekanizmaları

Güvenlik ataklarını algılamak, ataklardan korunmak veya geri dönmek için geliştirilmiş mekanizmalardır. Bir veya birden fazla güvenli mekanizması kullanılarak güvenli servisleri oluşturulur.

- Şifreleme
- Sayısal İmza
- Noterlik
- Erişim Kontrolü
- Veri Bütünlük Kontrolü
- Kimlik Değişimi
- Trafik Ekleme
- Yönlendirme Kontrolü

2.2 Saldırı Tespiti

Saldırı tespiti, bir sisteme yapılan izinsiz müdahalelerin mümkünse önceden, olay anında veya daha sonra; sistem günlüklerinden ve/veya bilgisayar ağı trafiğinden alınan verilerden yola çıkılarak çeşitli metotların yardımı ile analizidir. Saldırı tespitinin amacı bu müdahaleleri mümkünse önlemek ve saldırıları kayıt altına alarak e-posta, kısa mesaj, konsol uyarı mesajı v.b. gibi yollardan yetkili kişileri bilgilendirmektir. Korumak istediğimiz sistemi veya bilgisayar ağımızı bir eve benzetirsek saldırı tespitinin işlevi bu evdeki hırsız alarmının yaptığı göreve denk düşer. Evin kapısı güvenlik duvarı, kilidi şifreleme yöntemleri, kapıdaki göz merceği kimlik doğrulama sistemi olarak düşünülebilir. Saldırı tespiti evin içine hırsızın girmesini engelleyemez. Ancak hırsız tüm güvenlik önlemlerini aşarak eve girdiyse evin sahibini veya güvenlik güçlerini uyarmak için devreye girer. Eğer mümkünse ses ve video kaydı ile bu izinsiz girişi kaydederek ileriye yönelik hukuksal bir kanıt toplamaya çalışır. Saldırı tespiti tek başına asla tam bir güvenlik çözümü olamaz. Sistemimizin güvenliğini sağlamak için gereken şifreleme, doğrulama, kimlik kontrolü, güvenlik duvarı, erişim kontrolü vb. güvenlik önlemlerini aldıktan sonra ancak saldırı tespiti çözümlerine geçilmelidir. Kapısı sonuna kadar açık olan bir eve hırsız alarmı kurmak gereksizdir (korsan tuzağı (honeypot) sistemleri hariç).

Çizelge 2.2 Saldırı Tespit Kategorileri (Blomqvist & Skantze, 1995).

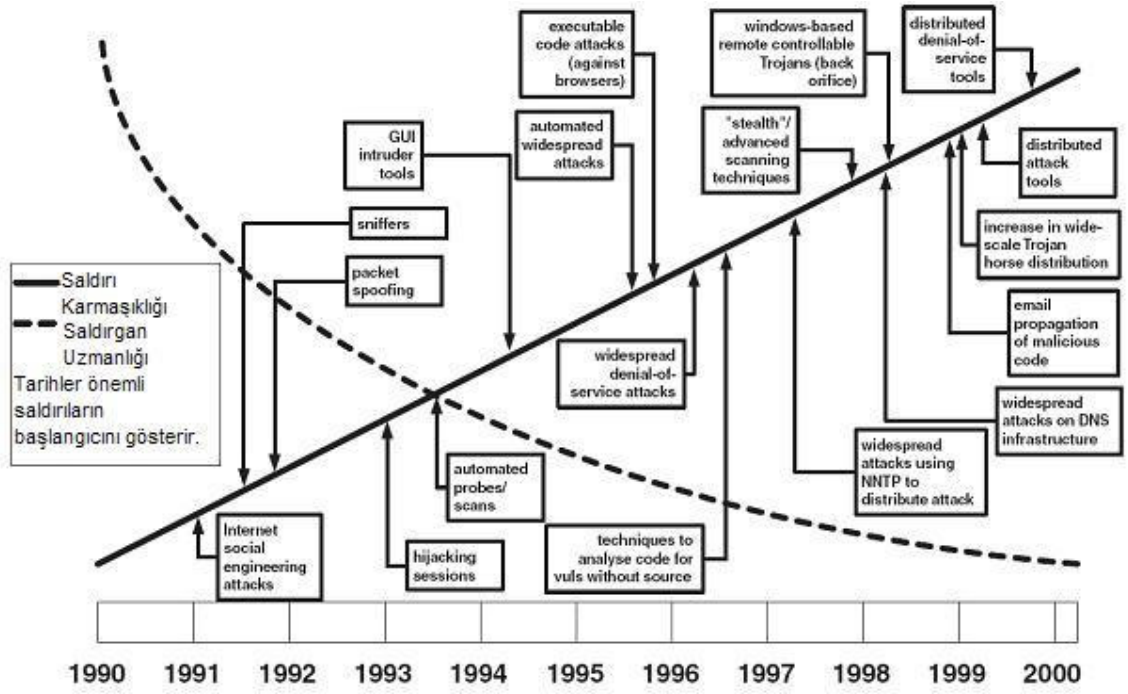
		Tespit	
		Evet	Hayır
Saldırı	Evet	Doğru (1)	Yanlış (2)
	Hayır	Yanlış (3)	Doğru (4)

Saldırı tespitinde yanlış tespit iki farklı şekilde olmaktadır. Saldırıyı yakalayamamak ve saldırı içermeyen bir bilgiyi saldırı olarak tespit etmek. Saldırı ve tespiti arasındaki ilişki Çizelge 2.2'de ifade edilmiştir. 1. ve 4. kategoriler saldırı tespit sisteminin doğru ve beklenen davranışlarıdır. Yakalanamayan doğru saldırılar (false-negative) (2. kategori) ideal bir saldırı tespit sisteminde minimum olmalıdır. Hatalı pozitif (false-positive) olarak nitelendirilen tespitler 3. kategoridedir. Hatalı pozitifler belirli bir sıklığa kadar kabul edilebilir.

2.2.1 Saldırı Tespit Sistemlerinin Gelişimi

James Anderson'un 1980 yılında yayımladığı "Computer Security Threat Monitoring and Surveillance"(Anderson, 1980) adlı teknik rapor, saldırıların incelenmesi ve tespiti konusunda ilk ciddi ve kapsamlı yayındır. Bu rapor, bilgisayarlardaki denetim günlüklerinden faydalanarak kötüye kullanımların ve tehdit sınıflarının belirlenebileceğini ortaya koymuş, denetim günlüklerinin bilgisayar güvenliği konusunda daha verimli nasıl kullanılabileceği konusunda önerilerde bulunmuştur (McHugh, 2001; McHugh, 2001; Schultz et al., 2003). 1970'lerde bilgisayar ağları bugünkü gibi yaygın olmadığı için bilgisayar ağı güvenliği de önemsenmiyordu. Bilgisayar sistemlerinin aynı anda birden fazla kullanıcıyı desteklemesi ile birlikte kullanıcı işlemlerinin ve verilerinin birbirlerine karışmadan çalıştırılması ve saklanması en büyük güvenlik sorununu oluşturdu. *Multics (Multiplexed Information and Computing Service)* [34], *EXEC 8* [33] gibi çok kullanıcıyı destekleyen işletim sistemlerinde birçok güvenlik ihlali ile sonuçlanabilecek zayıflıklar mevcuttu. 1980'lerin ortasına kadar bilgisayar ağı güvenliği hakkında kapsamlı bir çalışma yapılmamıştır. Bu dönemde en önemli sorun işletim sistemlerinin varsayılan ayarlarının değiştirilmeden aynen kullanılması sonucu oluşan zayıflıklardı. Çünkü varsayılan sistem şifresi her zaman aynı geldiği için varsayılan

ayarda kullanılan bir sisteme girmek çok basitti. Yine aynı dönemde PC'lerin (kişisel bilgisayar) ortaya çıkması ile bilgisayar virüsleri de yaygınlaşmaya başladı. Özellikle disket sürücüler ile taşınan virüsler birçok sisteme zarar verdi. 2 Kasım 1988 tarihi bilgisayar ağı güvenliğinde bir dönüm noktasıydı. Bu tarihte ilk defa tüm Internet'i kapsayan bir saldırı gerçekleşti. Robert T. Morris, kendi kendini yenileyen ve değişik iletişim yöntemleri ile kendini yayan bir programcık geliştirdi ve Internet'te yaydı. Morris kurdu, UNIX işletim sisteminde bulunan *sendmail* adlı programdaki bir açıktan faydalanıyordu. Bu saldırı o kadar etkili olmuştu ki DARPA bunun üzerine Carnegie Mellon Üniversitesi Yazılım Mühendisliği Enstitüsü'nde (SEI) CERT/CC' yi (Bilgisayar Acil Yardım Takımı) kurdu. 1990'lı yılların ikinci yarısında arabellek taşırma saldırıları büyük bir ivmeyle yaygınlaştı. Bu saldırı türünün yaygınlaşmasının sebebi Internet üzerindeki bilgisayarların neredeyse %25'inin arabellek taşırma saldırılarına karşı zayıf olması ve bunun sonucunda rasgele seçilen hedeflerde başarı oranının yüksek olmasıydı. Ayrıca bu saldırıya karşı zayıf olan programların çoğunluğu yönetici (administrator veya root) hakları ile çalıştığından saldırgan sistemin kontrolünü tamamıyla ele geçirebiliyordu. Bir başka avantajı da saldırı yazılımlarına ve tekniklerine Internet üzerinden rahatlıkla erişilebilmesi ve uygulama karmaşıklığının son derece düşük olmasıydı. Yıllar geçtikçe saldırıların karmaşıklığı artmasına rağmen, teknik ve yazılımlara erişimin kolaylaşması en tecrübesiz bilgisayar kullanıcısının bile hazır araçlar yardımı ile bir anda ciddi bir saldırgana dönüşmesine sebep oldu. Şekil 2.1'deki grafik CERT/CC'nin hazırladığı yıllara göre saldırı çeşitleri, karmaşıklığı ve saldırgan görüntüsünü göstermektedir.



Şekil 2.1 Yıllara göre saldırı karmaşıklığının gelişimi ve saldırgan becerisinin gerilemesi (McHugh, 2001).

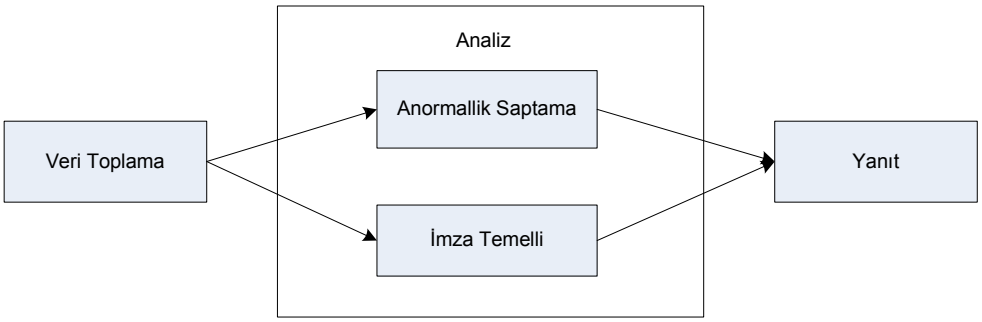
Yapılan saldırı sınıflandırma çalışmaları sonucunda ilginç bulgular elde edilmiştir. İlk bilgi güvenliği saldırılarının en büyük dayanak noktası olan hatalı veya varsayılan olarak yapılandırılan sistemler, gereğinden fazla veya yanlış yetkilendirme düzeyinde çalışan programlar hala günümüzdeki saldırıların ana nedendir (%58). Diğer zayıflıklar ise protokollerin tasarım eksikliklerinden (%10) ve uygulama hatalarından (% 2) kaynaklanmaktadır (McHugh, 2001).

2.2.2 Saldırı Tespit Yaklaşımları

Saldırı tespiti sinyal tespiti probleminin bir benzeri olarak görülebilir. Saldırılar normal sinyal içerisindeki gürültüler olarak düşünülebilir. Klasik sinyal tespiti yaklaşımında, hem sinyalin hem de gürültünün dağılımı bilinmektedir ve bir karar verme işlemi, verilen örneğin sinyal-artı-gürültü mü yoksa sadece gürültü mü olduğunu tespit etmektedir. Klasik sinyal algılayıcıları karar almada her iki türlü bilgiyi de (sinyal ve gürültü) kullanırlar ancak saldırı algılayıcıları karar alırken ya sinyal (imza tabanlı algılayıcılar) ya da gürültü (anormallik

tabanlı algılayıcılar) bilgisini kullanmaktadırlar. Her yaklaşımın kuvvetli ve zayıf yanları vardır. Her iki dağılımın da karakteristiğini çıkarmak başlıca problemdir (McHugh, 2001).

Stefan Axelsson'un 2000 yılında yayınladığı teknik raporunda (Axelsson, 2000) yaptığı tasnif, saldırı tespit sistemlerini analiz metotlarına göre sınıflandırır. Şekil 2.2'de saldırı tespitinin genel işleyişi gösterilmektedir.



Şekil 2.2 Saldırı tespit adımları.

Saldırı tespit sistemleri uygulandıkları yere ve kullandıkları algoritmik yaklaşıma göre dört başlıkta incelenebilirler.

- Sunucu tabanlı saldırı tespit sistemleri
- Ağ tabanlı saldırı tespit sistemleri
- Anormallik saptama temelli saldırı tespit sistemleri
- İmza temelli saldırı tespit sistemleri

2.2.2.1 Sunucu Tabanlı Saldırı Tespit Sistemleri

Sunucu temelli saldırı tespit sistemleri saldırı tespitinin ilk kullanım alanıdır. Ana bilgisayarlar üzerindeki denetim günlüklerini inceleyerek sisteme yapılan güvenlik ihlallerini veya olağandışı faaliyetleri tespit etmeye çalışırlar. Sisteme yapılan girişler, çıkışlar, dosya erişimleri ile sistemde çalıştırılan uygulamalar, uygunsuz bir şekilde sonlanan programlara ait bilgiler denetim günlüklerine eklenerek sunucu tabanlı saldırı tespit sistemlerinin girdilerini oluştururlar. Sunucu tabanlı saldırı sistemlerinin etkinliği, uygun bir sistem günlüğü ve denetim mekanizmasının oluşturulması ile doğru orantılıdır. Sistemdeki kullanıcı görüntüleri

ve yetkileri ayrıntılı ve kesin bir şekilde tanımlanmalıdır. Sistemde çalışan her programın doğru ve bilgilendirici bilgiyi denetim günlüklerine yazması önemlidir. Bir başka nokta da sınırlı kapasitedeki depolama alanında sistem günlüklerinin ne kadar ayrıntılı tutulacağını belirlemek gerekmektedir. Ayrıntılı bir denetim günlüğünü incelemek için harcanan işlemci zamanı sistemi olumsuz etkilemesi bir yana saldırı tespit sisteminin çok fazla veri arasından gerçek saldırıları bulmasını da zorlaştırabilir. Bu durumda sistem günlüklerinin oluşturulmasına ayrı bir özen göstermek gerekmektedir. Sunucu tabanlı sistemler gerçek zamanlı veya biriktirilmiş veriler üzerinde çalıştırılabilir. Gerçek zamanlı sistemlerin avantajı saldırıyı olduğu anda yakalayabilmesi ve sisteme olan zararı minimuma indirebilmesidir. Ancak, gerçek zamanlı sistemlerde bilgisayarın genel başarımını olumsuz etkilenir. Biriktirilmiş veriler üzerinde çalışan sistemlerde denetim günlükleri ya uygulamanın bulunduğu bilgisayarda ya da merkezi başka bir bilgisayarda toplanır. Bu yöntemde günlükleri ayrıntılı bir şekilde analiz etme fırsatı vardır. Ancak, tespit işlemi saldırıdan çok sonra yapıldığı için saldırıya karşı sadece sistemi kurtarma veya saldırganı tespit işlemi yapılabilir. Bir başka nokta da saldırgan eğer sistemi ele geçirdiyse pek tabii olarak denetim günlüklerini de değiştirmiş veya silmiş olabilir. Bu yüzden denetim günlüklerini güvenli bir ortamda saklamak veya merkezi ve güvenli başka bir sunucuya yönlendirmek gerekmektedir. Sunucu tabanlı saldırı tespit sistemlerinin avantajları:

- Bir saldırının başarısını veya başarısızlığını büyük bir doğrulukla belirleyebilir.
- Sistem aktivitelerini izleyerek sistem hakkında çok daha ayrıntılı bilgi edinebilir.
- Gerçek zamana yakın hızda saldırıyı tespit edebilir ve yanıtlayabilir.
- Ek bir donanım veya maliyet gerektirmez.

2.2.2.2 Ağ Tabanlı Saldırı Tespit Sistemleri

Ağ saldırı tespit sistemleri bilgisayar ağı trafiğini inceleyerek bilgisayar ağından gelebilecek saldırıları sezen yapılardır. Analiz verisi olarak sistem günlükleri yerine bilgisayar ağı haberleşmesinde iletilen verileri kullanır. Bu yaklaşımın sunucu yaklaşımına göre bir avantajı tek bir sistem ile birden fazla bilgisayar sisteminin korunmasını sağlayabilmesidir. Tüm bilgisayar ağı trafiğini dinleyebilecek konumda ve yapılandırmada (promiscuous mode) kurulan sistemler aynı anda birden fazla bilgisayar sistemine hizmet verebilir. Ayrıca, saldırı henüz ilk aşamadayken saldırıyı tespit edip bağlantıyı koparark, güvenlik duvarına uygun

kuralları ekleyerek veya anahtarlama cihazlarındaki yapılandırma ayarlarını değiştirerek saldırının oluşumunu engelleyebilir. Sunucu tabanlı sistemlere göre dezavantajı ise konsol mesajlarını, aramalı bağlantıları ve uygulama temelli hataları tespit edememesidir. Ayrıca şifreli bir şekilde iletilen verinin analizi de güç veya imkânsız olmaktadır. Buna karşın kolay gerçeklenmesi ve uygulanması, platform bağımsız oluşu nedeni ile yaygın bir şekilde kullanılmaktadır. Ağ tabanlı saldırı tespit sistemlerinin bir başka kısıtlandığı nokta ise bilgisayar ağı trafik hızının günümüzde hızlı bir şekilde artması ve bu hızlı bilgisayar ağı üzerinden akan trafiğin tümünün analizinin zor olmasıdır. Bu soruna çözüm, bilgisayar ağı cihazlarına iliştirilmiş, donanım seviyesine daha yakın uygulamalar yaparak işlem hızını arttırmaktır ancak bu sistemlerin satın alma maliyeti oldukça fazladır. Bir başka yöntem ise analiz işlemini basamaklandırarak veri üzerinde ön eleme yapmak ve daha güvenilir olduğunu varsaydığımız verileri daha yalın bir işlemden geçirerek analiz etmektir. Ağ saldırı tespit sistemleri saldırının hedefi de olabilirler. TCP/IP protokolünün işletim sistemleri arasındaki gerçeklenmesindeki farklılıklar sistemin saldırgana verdiği yanıtları farklılaştıracağından saldırgan tespit sisteminin farkına varabilir. Ağ saldırı tespit sistemlerinin bir çeşidi de durum bilgili ağ saldırı tespit sistemleridir. Durum bilgili ağ saldırı tespit sistemlerinde TCP oturumlarının durum bilgileri (açılma, ilerleme, yeniden iletim, kapanma vb.) incelenerek sistem hakkında daha bütünsel bir bilgi elde edilebilmektredir. Durum bilgili ağ saldırı tespit sistemlerinde saldırgan, içeriği değiştirilmiş paketler göndererek sistemin durum bilgisi oluşturmasını engelleyebilir ve sistemi hataya zorlayabilir (Ptacek & Newsham, 1998).

2.2.2.3 Anormallik Saptama Temelli Saldırı Tespit Sistemleri

Bu sistemler normal aktiviteler ile saldırı aktivitelerinin belirli bir inceleme ile birbirinden ayrılabileceği düşüncesi ile geliştirilmişlerdir. Anormallik tespitinde bilinen saldırılar değil veri trafiğindeki olağandışlılıklar gözlemlenir. Olağandışı bir aktivite şüpheli olarak kabul edilir. Anormallik saptama temelli bir algılayıcının tasarlanmasında öncelikle incelenen ortam, kullanıcı veya sistem için “normal” aktivitenin ne olduğunun ayrıntılı ve doğru bir şekilde belirlenmesi gerekmektedir. Normal aktivitenin belirlenmesi el ile yapılabildiği gibi otomatik olarak kullanıcı görünüşü ve normal trafik analizi ile de çıkarılabilir. Gerçekleşen bir aktivitenin şüpheli bir aktivite olması için normal aktiviteden belirli bir oranda farklılaşması gerekmektedir. Bu oranların doğru tespiti sistemin doğruluğu ve yanlış pozitif oranının minimum olması için önemlidir. Anormallik tabanlı saldırı tespit sistemleri;

- Bilinmeyen saldırıları yakalayabilir
- Saldırıları ayrıntılı bir biçimde tanımlayamaz
- Bilinen saldırıları kaçırabilir
- Yanlış pozitif oranı yüksektir
- "Normal" kavramı kullanıcıdan kullanıcıya ve sistemden sisteme değiştiği için gerçeklenen bir uygulamanın diğer sistemlere adapte edilmesi zordur.

Çizelge 2.3'te anormallik tabanlı saldırı tespit sistemleri kullanılan model ve metotlara göre örneklenerek sınıflandırılmıştır.

Çizelge 2.3 Anormallik Saptama Temelli Saldırı Tespit Algılayıcıları (Axelsson, 2000).

Stil	Model	Metot	Örnekler
Kendi Kendine Öğrenen	Zamansız Seriler	Kural Modeli	W&S
		Tanımlayıcı Verileri Derleme	IDES NIDES EMERALD JiNao Haystack
	Zaman Serileri	Sinir Ağları	Hyperview
Programlanmış Sistemler	Tanımlayıcı Verileri Derleme	Basit Veri Derleme	MIDAS NADIR Haystack
		Basit Kural Tabanlı	NSM
		Eşik Değeri	
	Varsayılan Olarak Ret	Durum Serileri Modeli	DEPM JANUS Bro

Kendi Kendine Öğrenen Sistemler: Kendi kendine öğrenen sistemler kuruldukları aşamada "normal" verilerden oluşan veri trafiğini uzun bir süre inceleyerek aşağıda belirtilen yöntemlere göre bir sonraki tanıma ve tespit aşaması için bir model oluştururlar (Axelsson, 2000).

- **Zamansız Seriler (Non-time Series):** Sistemin normal davranışını modellemek için zaman serilerini kullanmayan bir seçme algoritması uygulayan algılayıcılardır.

    - **Kural Modeli (Rule Modeling):** Sistemin kendisi trafiği öğrenir ve öğrendiği bilgiyi sistemin normal işleyişini formüle eden kurallara dönüştürür. Tespit

aşamasında sistem kuralları uygular ve eğer kural veritabanı ile incelenen trafik arasında zayıf bir eşleşme varsa alarm verilir.

- **Tanımlayıcı Verileri Derleme (Descriptive Statistics):** Sistem, kullanıcı veya bilgisayar ile ilgili belirleyici, basit ve lineer parametreleri toplayarak bir görüntü oluşturur. Oluşturduğu görüntü ve incelenen trafiğin uzaklık vektörünü oluşturarak arasındaki uzaklığı ölçer. Eğer, uzaklık belirli bir eşik değerinden fazla ise alarm verilir.

- **Zaman Serileri (Time Series):** Bu model çok daha karmaşık yapılar içeren zaman serilerini kullanır. Örnek teknikler saklı markov modeli ve yapay sinir ağlarıdır.

  - **Yapay Sinir Ağları (YSA):** Yapay sinir ağları "kara kutu" modelleme yaklaşımının bir örneğidir. Öncelikle "normal" trafik YSA sistemine verilerek sistemin normal trafik özelliklerini dolaylı yoldan tanıması sağlanır. Öğrenme aşamasından sonra incelenecek trafik YSA sistemine verilerek, sistemin karar verilmesi sağlanır. Ancak, yapılan çalışmalar sonucu YSA sisteminden alınan sonuçların tek başına yeterli kalitede sonuçlar vermediği görülmüştür. Bu yüzden YSA'den çıkan verileri ikinci bir uzman sisteme girdi olarak vererek son karar oluşturulur.

Programlanmış Sistemler: Programlanmış sistemlerde sistemin öğretilmesi için bir kullanıcıya ihtiyaç vardır. Kullanıcı veya kullanıcılar sistemi programlayarak sistemin belirli kural dışı olayları algılamasını sağlar. Sistemi düzenleyen kullanıcı oluşan aktivitenin güvenlik ihlali alarmı olarak algılanıp algılanamayacağına dair kararı oluşturur.

- **Tanımlayıcı Verileri Derleme (Descriptive Statistics):** Bu sistemler, bilgisayar sistemindeki bazı parametrelerin istatistiksel değişimlerini gözlemleyerek normal istatistiksel davranışın görüntüsünü oluştururlar. Kullanılan parametreler, sisteme hatalı girişler (sayısı, sıklığı ve zamanı), açılan bilgisayar ağı bağlantılarının sayısı (zamana göre veya bir **t** anında), hatalı komutların sayısı v.b. olabilir.

- **Basit Veri Derleme (Simple Statistics):** Bu metotta toplanan istatistikler daha özet bir saldırı tespit kararı verebilmek için üst seviye bileşenler tarafından değerlendirilir.

- **Basit Kural Tabanlı (Simple Rule-Based):** Kullanıcı toplanan istatistiklerin uygulanması için sisteme basit fakat bileşik kurallar verir.

- **Eşik Değeri (Threshold):** Programlı – Tanımlayıcı verileri derleme sistemlerindeki en basit örnek eşik değeri belirleme yöntemidir. Sistem gerekli istatistikleri topladıktan sonra kullanıcı ön tanımlı eşik değerlerini programlayarak saldırı alarmın nerede çalacağına karar verir. Örnek olarak; "üç adet başarısız girişten sonra alarm ver" gibi.

- **Varsayılan Olarak Reddetmek (Default Deny):** Bu yaklaşımda sistemin tüm durumları ve işlemleri güvenlik anlayışına göre, kesin olarak belirlenmelidir. Daha sonrasında tüm faaliyet ve durumlar varsayılan şekilde saldırı olarak işaretlenmelidir. Yapılan güvenlik politikası ile sistemin işleyişi arasında sıkı bir uyuşma olmalıdır. Genelleme yapmak yerine olabildiğince ayrıntılı kurallar oluşturmak politika ile sistem arasındaki uyumu arttırmaktadır.

    - **Durum Serileri Modeli (State Series Modeling):** Durum serileri modelinde güvenlik politikası durum serileri olarak kodlanmıştır. Durumlar arası geçişler, uzman sistemlerde kullanılan durum serileri gibi kesin ve doğrudan değil dolaylıdır. Sonlu otomata bir durumu yakaladığında bir sonraki duruma geçilinceye kadar bekler. İzlenen eylem uygunsa bir sonraki duruma geçilir. Eğer mevcut durumdaki eylem kesinlikle o duruma uygun değilse alarm verilir. Gözlenen eylemler genellikle dosya erişimi (okuma, yazma), bağlantı noktası açma gibi güvenlikle ilgili eylemlerdir.

2.2.2.4 İmza Temelli Saldırı Tespit Sistemleri

İmza temelli saldırı tespit sistemleri saldırı tespiti için saldırılara ait örüntü bilgisinden yararlanırlar. Virus koruma programlarında olduğu gibi imza temelli saldırı tespit sistemlerinin her saldırıya ait ayırt edici bilgiler içeren veritabanı vardır. İmza temelli saldırı

tespit sistemlerinin bir saldırıyı algılaması için o saldırı hakkında yeterli derecede bilgi sahibi olması gerekir. İmza temelli saldırı tespitinde bilgisayar ağı üzerinde akan arka plan trafiğin normal veya anormal olması dikkate alınmaz. Sadece gelen paketlerin bilinen saldırılar ile uyuşup uyuşmadığı kontrolü yapılır. Saldırı bilgisi bir bilgisayar ağı paketindeki özgül bir örüntü olabileceği gibi saldırı bilgisini betimleyen sonlu otomata veya yapay sinir ağı olabilir. İmza temelli saldırı tespiti yönteminde bilinen ataklar büyük bir başarı oranı ile yakalanır. Eğer saldırı bilgisinde uygun bir soyutlama yapılırsa o saldırıya benzerlik gösteren yeni saldırılar yakalanabilir. Saldırı bilgisi fazla ayrıntılı tanımlanırsa ise bu sefer de ufak değişimler kaçırılabilir. Ancak, tümüyle yeni bir saldırı çeşidini kesinlikle yakalayamazlar. Sistem saldırı karakteristiklerini bildiği için gerçekleşen saldırılar hakkında ayrıntılı bilgi sahibi olunur. Bu yöntemin dayanak noktası saldırı hakkında bilgi edinmek olduğundan yeni çıkan her saldırı türü için sistem güncellenmelidir. Hem bilinen saldırılara karşı etkinliği hem de uygulamadaki kolaylığı nedeni ile günümüzde saldırı tespit sistemi ürünlerinin çoğunluğu imza temelli yaklaşımı kullanmaktadır. Çizelge 2.4'te imza tabanlı saldırı tespit sistemleri kullanılan model ve metota göre örneklenerek sınıflandırılmıştır.

Çizelge 2.4 İmza Temelli Saldırı Tespit Algılayıcıları (Axelsson, 2000).

Stil	Model	Metot	Örnekler
Programlanmış Sistemler	Durum Modeli	Durum Geçişi	USTAT
		Petri Ağı	IDIOT
	Zaman Serileri	Sinir Ağları	Hyperview
	Uzman Sistem		NIDES EMERALD MIDAS DIDS
	Dizgi Eşleme		NSM
	Basit Kural Tabanlı		NADIR ASAX Bro JiNao Haystack

Programlanmış Sistemler: Programlanmış sistemler kesin bir şekilde belirtilmiş karar kuralları vasıtasıyla programlanır. Karar kuralları basit yapıdadır ve oluşan bir saldırıdan neler beklendiğini açık bir şekilde ifade eder. Açık bir şekilde belirlenen saldırılar dışındaki trafiğe varsayılan olarak izin verilir. Bu nedenle güvenlik politikası açık bir nokta kalmayacak şekilde netleştirilmeli ve olabildiğince genel kurallar oluşturulmaya çalışılmalıdır.

- **Durum Modeli (State Modeling):** Durum modeli saldırıyı birkaç farklı durumdan oluşan bir yapı olarak ifade eder. Durum modelinde kullanılan iki yöntemden biri durum geçişi yöntemidir. Bu yöntemde saldırı tespitinde incelenen olay durumlar arası zincirleme bir şekilde kat ederek sonuca ulaşır. Olay son duruma gelirse saldırı alarmı verilir. İkinci yöntem ise petri ağlarıdır. Bu yöntemde durumlar sadece kuyruk yapısında değil ağaç yapıları gibi daha karmaşık ve belirli parametrelere göre farklı durumlara dallanabilecek şekilde oluşturulabilir.

- **Uzman Sistem (Expert System):** Uzman sistemler saldırı davranışını tanımlayarak sistemin güvenlik durumunu belirler. Genellikle ileri yönde ilerleyen durum serileri ve uygulamalara göre özelleşmiş araçlar kullanılır. Uzman sistemler yeterli derecede güçlü ve esnek bir sistem oluşturur. Ancak, daha basit metotlara göre işlem hızı daha düşüktür.

- **Dizgi Eşleme (String Matching):** Dizgi eşleme, alt dizgileri sistemler arasında iletilen bilgide veya sistemin ürettiği mesajlarda arayan basit, genellikle büyük küçük harfe duyarlı bir yöntemdir. Bu metot esnek bir yapıda değildir ancak basit, anlaşılır yapısı ve bilinen tehditlere karşı etkinliği nedeni ile birçok saldırı tespiti uygulamasında tercih edilmektedir.

- **Basit Kural Tabanlı (Simple Rule-Based):** Bu yöntem uzman sistemlere benzemek ile birlikte daha az ayrıntı ve karmaşıklık içerir. Daha hızlı işlem zamanları hedeflendiğinde bu yöntem kullanılabilir.

2.2.2.5 Diğer Sınıflandırma Yaklaşımları

Saldırı tespit sistemleri birkaç farklı özelliğe bağlı olarak sınıflandırılabilirler. Saldırı tespitinde analiz aşaması gerçek zamanlı, beklemeli veya toplu olarak yapılabilir. Tespitin zamanlaması sistemin başarım, saldırıyı önleme yetisi gibi etkenleri doğrudan etkiler. Sunucu temelli ve sistem günlüklerinden faydalanan sistemler genellikle beklemeli veya toplu analiz yaklaşımını benimserler. Bu hem sistem kaynaklarının verimli kullanımını hem de analizin ayrıntılı yapılabilmesini sağlar. Ağ temelli saldırı tespit sistemlerinde ise gerçek zamanlı analiz yaklaşımları saldırıyı oluşum anında önlemek ve sisteme vereceği hasarı minimuma

indirmek amacıyla tercih edilir. Gerçek zamanlı yaklaşımda verinin analizi zaman kısıtı nedeniyle detaylı bir şekilde yapılamaz.

Bir başka kategori de verinin işleniş tarzıdır. Veri sürekli bir şekilde veya toplu olarak işlenebilir. Bu kategori bir önceki kategori ile benzerlik gösterse de gerçek zamanlı analizde veriler toplu inceleneceği gibi gerçek zamanlı olmayan bir analizde veriler bir akış olarak da incelenebilir.

Saptanan saldırılara karşı verilen yanıt pasif veya aktif olabilir. Pasif yanıt sistemleri yetkili kullanıcıyı uyaran mekanizmalar kullanırlar. Saldırıyı sonlandırmak veya engellemek için bir çaba içerisine girmezler. Aktif yanıt sistemlerinin ise saldırıya karşı önleme mekanizmaları vardır. Aktif yanıt sistemleri kendi arasında ikiye ayrılır. Saldırıya uğrayan sistem üzerinde kontrol sağlayan aktif yanıt sistemleri bağlantıyı sonlandırma, zayıflığa sahip işlemi sonlandırma, güvenlik seviyesini arttırma veya güvenlik duvarını yeniden yapılandırma gibi işlemlerle saldırıyı önlemeye çalışırlar. Bir diğer yaklaşım ise saldırıyı saldırgan makinede önlemektir. Atak yapan sistemin işleyişini engellemek olan bu metodun gerçeklenmesi diğerine göre çok daha zordur.

Verinin toplandığı yer açısından merkezi veya dağıtık sistemler mevcuttur. Veriler tek bir merkezden toplanabileceği gibi birden fazla kaynaktan toplanıp da işlenebilir. Kullanılan algılayıcı tek olabileceği gibi birden fazla algılayıcı yardımıyla veriler de toplanabilir.

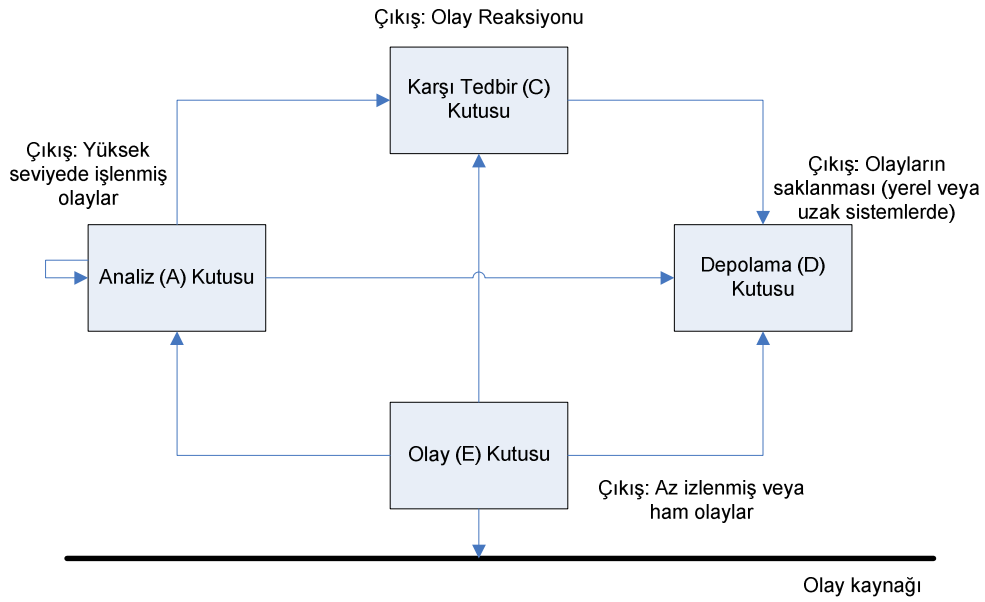
Saldırı tespit sistemlerinin kendilerine yönelen saldırılara karşı dayanıklılığı da bir başka sınıflandırma kıstasıdır. Ayrıca saldırı tespit sistemlerinin diğer sistemlerle uyumu ve birlikte çalışabilirliği de bir başka ölçüttür. Sistem karakteristiklerinin karşılaştırmalı genel tablosu Çizelge 2.5'de verilmiştir.

Çizelge 2.5 Saldırı Tespit Sistemlerinin Sistem Karakteristiklerine Göre Sınıflandırılması (Blomqvist & Skantze, 1995).

Sistem	Yıl	Tespit Zamanı	Veri Boyutu	Veri Kaynağı	Yanıt Tipi	Veri İşleme	Veri Toplama	Güvenlik	Uyum
Haystack	1988	Beklemeli	Yığın	Sunucu	Pasif	Merkezi	Merkezi	Düşük	Düşük
MIDAS	1988	Gerçek Zamanlı	Sürekli	Sunucu	Pasif	Merkezi	Merkezi	Düşük	Düşük
IDES	1988	Gerçek Zamanlı	Sürekli	Sunucu	Pasif	Merkezi	Dağıtık	Düşük	Düşük
W&S	1989	Gerçek Zamanlı	Sürekli	Sunucu	Pasif	Merkezi	Merkezi	Düşük	Düşük
Comp-Watch	1990	Beklemeli	Yığın	Sunucu	Pasif	Merkezi	Merkezi	Düşük	Düşük
NSM	1990	Gerçek Zamanlı	Sürekli	Ağ	Pasif	Merkezi	Merkezi	Düşük	Düşük
NADIR	1991	Beklemeli	Sürekli	Sunucu	Pasif	Merkezi	Dağıtık	Düşük	Düşük
Hyperview	1992	Gerçek Zamanlı	Sürekli	Sunucu	Pasif	Merkezi	Merkezi	Düşük	Düşük
DIDS	1992	Gerçek Zamanlı	Sürekli	Hepsi	Pasif	Dağıtık	Dağıtık	Düşük	Düşük
ASAX	1992	Gerçek Zamanlı	Sürekli	Sunucu	Pasif	Merkezi	Merkezi	Düşük	Yüksek
USTAT	1993	Gerçek Zamanlı	Sürekli	Sunucu	Pasif	Merkezi	Merkezi	Düşük	Düşük
DPEM	1994	Gerçek Zamanlı	Yığın	Sunucu	Pasif	Dağıtık	Dağıtık	Düşük	Düşük
IDIOT	1994	Gerçek Zamanlı	Sürekli	Sunucu	Pasif	Merkezi	Merkezi	Düşük	Yüksek
NIDES	1995	Gerçek Zamanlı	Sürekli	Sunucu	Pasif	Merkezi	Dağıtık	Düşük	Yüksek
GrIDS	1996	Beklemeli	Yığın	Hepsi	Pasif	Dağıtık	Dağıtık	Düşük	Düşük
CSM	1996	Gerçek Zamanlı	Sürekli	Sunucu	Aktif	Dağıtık	Dağıtık	Düşük	Düşük
Janus	1996	Gerçek Zamanlı	Sürekli	Sunucu	Aktif	Merkezi	Merkezi	Düşük	Düşük
JiNao	1997	Gerçek Zamanlı	Yığın	Sunucu	Pasif	Dağıtık	Dağıtık	Düşük	Düşük
EMERALD	1997	Gerçek Zamanlı	Sürekli	Hepsi	Aktif	Dağıtık	Dağıtık	Normal	Yüksek
Bro	1998	Gerçek Zamanlı	Sürekli	Ağ	Pasif	Merkezi	Merkezi	Yüksek	Düşük

2.3 Saldırı Tespit Sistemleri

Günümüzde çok sayıda ve farklı yaklaşımlarda saldırı tespit sistemleri geliştirilmektedir. Saldırı tespit sistemlerinin genel bir yapısını oluşturmak bu konudaki çalışmalara yardımcı olacaktır. Genel saldırı tespit çatısı (The Common Intrusion Detection Framework (CIDF)) birbirleriyle bağlantılı farklı bileşenlerden oluşur. Bu bileşenler, olay üreteçleri ("E-kutuları"), analiz motorları ("A-kutuları"), depolama mekanizmaları ("D-kutuları") ve karşı tedbirdir ("C-kutuları"). E-kutularının görevi olaylar hakkında sisteme bilgi girdisi sağlamaktır. E-kutuları, sistemin algılamayı sağlayan kısmıdır. A-kutuları, olay üreteçlerinden aldıkları veriyi anormallik tespiti, imza tabanlı, grafik analizi v.b. yöntemlerini kullanarak analiz ederler. Her yöntemin diğerlerine göre avantaj ve dezavantajları bulunmaktadır. Kullanılacak analiz yaklaşımının seçimi sistemin başarısını etkileyen en önemli faktördür. E-kutuları ve A-kutuları büyük miktarlarda veri üretebilir. Bu verinin gerektiğinde erişilmesi ve kullanılması için uygun bir şekilde saklanması gerekmektedir. D-kutuları veri saklama işini üstlenmektedir. Tespit edilen saldırıya karşı alarm vermek veya bir tedbir almak gerekmektedir. C-kutuları saldırılara karşı yetkili kullanıcıları uyarır, saldırgan ile olan ağ bağlantısını keser; güvenlik duvarı, yönlendirici ve anahtarlama cihazlarındaki kuralları saldırıyı önleyecek şekilde değiştirir.



Şekil 2.3 CIDF bileşenleri ve birbirleriyle ilişkileri (Ptacek & Newsham, 1998).

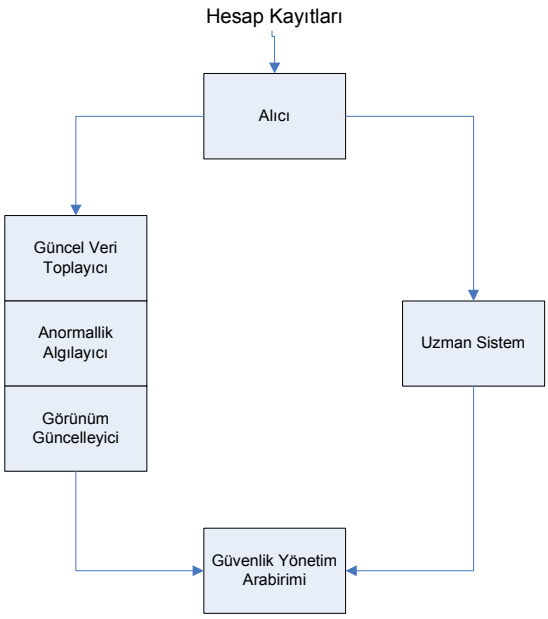
Bu bölümde aşağıda belirtilmiş olan başlıca saldırı tespit sistemlerinden bahsedilmektedir.

- IDES ve NIDES
- Haystack
- MIDAS
- Hyperview
- USTAT ve NETSTAT
- Bro
- Snort

2.3.1 IDES ve NIDES

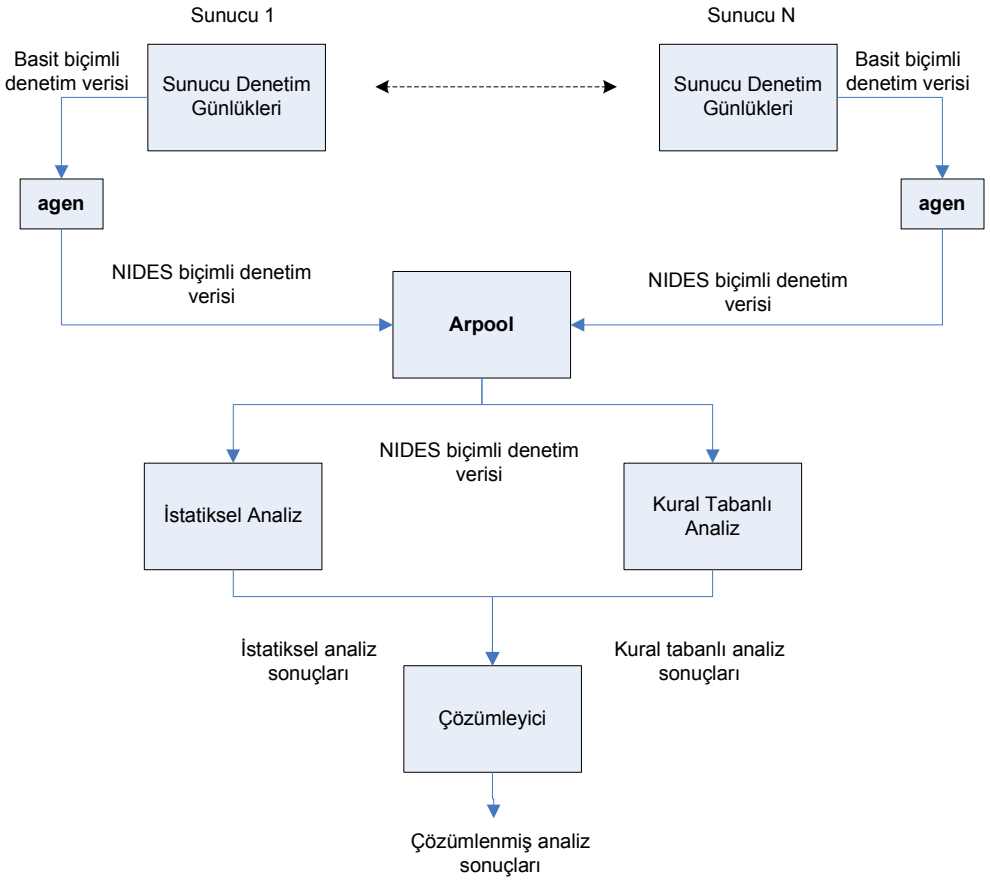
IDES (Intrusion Detection Expert System) (Lunt, 1989), gerçek zamanlı çalışan bir saldırı tespit sistemidir. İlk ve en iyi dokümante edilmiş saldırı tespit sistemlerinden biridir. Günümüzde IDES sistemi güncelliğini yitirmiştir. IDES sistemi, NIDES (Next Generation Intrusion Detection Expert System) projesi ile birleştirilmiş olarak sürdürülmektedir.

IDES'in temel motivasyonu, kullanıcı davranışlarının öngörülebilir ve kararlı bir yapıda olması ve farklı istatistiksel yöntemler yardımı ile özetlenebilir olmasıdır. Gerçekleşen eylemler kullanıcı davranışlarından özetlenen bilgiler ile ilişkilendirilerek ortaya çıkan sapmalar muhtemel saldırgan davranışlar olarak işaretlenmektedir. IDES her yeni denetim kaydını sisteme girildiği anda ve kullanıcı oturumunun sonunda işler. Denetim kaydının bilinen görünümlere yakınlığı incelenerek uygun kullanıcı grubu ile ilişkilendirilir. IDES analiz için iki temel görünüm kullanmaktadır. İlki çalışma günlerindeki kullanıcı davranışlarını modellerken ikincisi tatil günlerindeki kullanıcı davranışlarını dikkate alır. Çalışma günlerinde yoğun olarak kullanılan sistem tatil günlerinde seyrek veya hiç kullanılmadığı için bu iki durumu ayrı incelemek kullanıcı davranışlarının daha iyi modellenmesini sağlamaktadır. Bu ayırım sistemin "normal" davranışın daha iyi tanımlanmasını sağlar ve doğruluk oranını arttırır.



Şekil 2.4 IDES genel yapısı (Lunt, 1989).

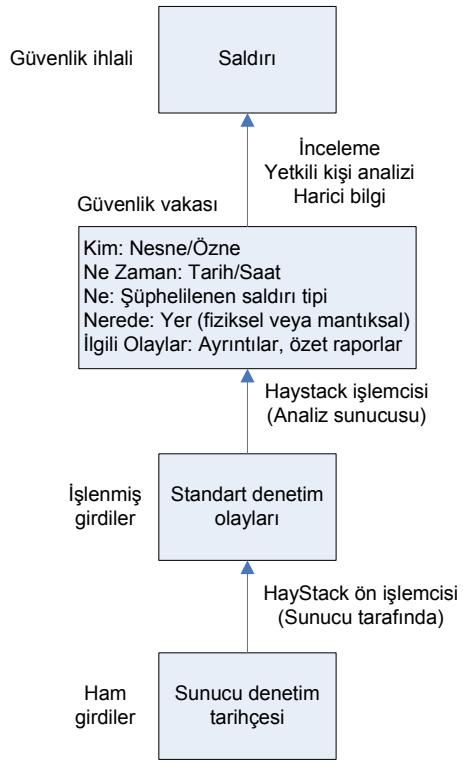
NIDES (Next Generation Intrusion Detection Expert System) (Anderson et al., 1995) , IDES sisteminin devamı niteliğindedir. Modüler bir yapıdadır. Birçok sistemden toplanan denetim günlükleri gerekli dönüşümlerden geçirildikten sonra istatistiksel ve kural tabanlı bir uzman sistem ile analiz edilir. Şekil 2.5'te gösterildiği üzere, analiz edilen her sunucuda bir agen işlemi çalışmaktadır. Agen sunucudan aldığı basit biçimli denetim verisini NIDES formatına çevirir ve Arpool'a yollar. Arpool sadece NIDES makinesinde çalışır ve tüm sunuculardan gelen denetim verisini tek bir denetim verisine dönüştürerek istatistiksel ve kural tabanlı analiz bileşenlerine iletir. Analiz sonuçları çözümleyici tarafından değerlendirilir ve bir alarm durumunun oluşup oluşmadığı belirlenerek kullanıcı ara yüzüne ve uyarı raporlama mekanizmalarına (örn. e-posta, kosol mesajı vb.) iletilir.



Şekil 2.5 NIDES'in işleyiş grafiği (Anderson et al., 1995).

2.3.2 Haystack

ABD Hava Kuvvetleri'ndeki çok kullanıcılı bilgisayar sistemleri için geliştirilmiştir. Sunucu tabanlıdır. Anormallik tespiti ve İmza denetimi yaklaşımlarını birlikte kullanır. Anormallik tespiti iki kavram üzerine oturtulmuştur; her kullanıcı için bir model oluşturulur ve kullanıcıların geçmiş davranışlarına bakılarak şu anki davranışları ile karşılaştırılır. Diğer kavramda ise daha ön tanımlı kullanıcı grupları kullanılır ve grup davranışları incelenir. Her iki kavramın birleşimi saldırı tespiti için sisteme yeterli bilgi sunmaktadır.



Şekil 2.6 Haystack işleyişi (Smaha et al., 1988).

2.3.3 MIDAS

MIDAS, ABD Ulusal Bilgisayar Güvenliği Merkezi tarafından geliştirilmiştir (Sebring et al., 1988). MIDAS, buluşsal (heuristic) yaklaşımı kullanır. Bilgisayar güvenlik operatörünün bir saldırı izini ararken kullandığı yöntemleri örnek alır. İki seviyede sistem günlüklerini inceler. İlk seviye anlık durumları kaydeder (örn. Hatalı girişlerin sayısı). İkinci seviye ise ilk seviyeden aldığı verileri bir bütün olarak değerlendirir ve bir saldırı oluşup oluşmadığını tespit eder. Örneğin, kullanıcı bir oturum boyunca 40 adet hatalı komut girmişse ve *suid*, *priv* komutlarını çalıştırmaya teşebbüs etmişse ve oturum olağandışı bir saatte açılmışsa bu kullanıcının bir saldırgan olma ihtimali yüksektir.

2.3.4 Hyperview

Hyperview, uzman sistem ve yapay sinir ağı (YSA) bileşenlerinden oluşmaktadır. Uzman sistem, denetim tarihçesinde bilinen saldırılara ait izleri arar. YSA ise kullanıcı davranışlarını

öğrenir ve eğer denetim tarihçesi, öğrenilmiş davranışlardan sapma gösterirse uyarı verir. Denetim tarihçesi çok değişkenli zaman serileri şeklinde YSA' ya verilmektedir. YSA çıktıları belirli derecelerde ağırlandırılarak YSA' ya tekrar girdi olarak verilmektedir. Böylece YSA olayların tarihçesini tutarak kullanıcı davranışlarını öğrenmektedir. YSA'ya iki adet uzman sistem bağlanmıştır. İlk uzman sistem YSA'nın öğrenmesini ve işleyişini kontrol eder. Örneğin, sistemin anormal davranışı öğrenmesini engeller ve YSA çıktılarını değerlendirir. İkinci uzman sistem bilinen saldırı örüntülerinden yararlanarak denetim tarihçesini inceler. İlk uzman sistemden gelen (dolayısıyla YSA'dan) çıktıları da birlikte değerlendirerek bir alarm durumu oluşup oluşmadığına karar verir (Debar et al., 1992).

2.3.5 USTAT ve NetSTAT

USTAT (State Transition Analysis for UNIX) (Ilgun, 1992) ve NetSTAT (Network Based State Transition Analysis) (Vigna, 1999) sistemleri saldırı tespiti için durum geçiş analizi yöntemini kullanmaktadırlar. Durum geçiş analizinde, durumlar sistemdeki veya bilgisayar ağındaki olaylardan meydana gelmektedir ve olaylar arasındaki geçişler ile saldırılar tanımlanmaktadır. Tanımlanan saldırı durum bilgisi USTAT tarafından sistem günlükleri ile karşılaştırılmaktadır. Saldırının USTAT tarafından tespit edilebilmesi için iki ön şartın oluşması gerekmektedir. Saldırı sistem üzerinden görülebilir bir etki bırakmalıdır ve bu etki dıştan gelen bir bilgiye ihtiyaç olmadan fark edilebilmelidir. Örneğin, bilgisayar ağı pasif olarak dinlendiğinde USTAT bu durum hakkında yeterli bilgi toplayamadığı için tespit de edemez. Kendini maskeleyen (yetkili bir kullanıcı olarak gözüken) bir saldırganın sistem üzerinde normal davranış gösterdiği sürece yakalanması çok güçtür. Ancak, yüksek yetki seviyesine erişmek gibi şüpheli bir faaliyette bulunduğunda USTAT saldırıyı yakalayabilir. NetSTAT, USTAT'tan farklı olarak, durum geçiş diyagramlarını gerçek zamanlı bilgisayar ağı verisi ile oluşturur.

2.3.6 Bro

Bro, gerçek zamanlı ağ saldırı tespit sistemidir (Paxson, 1988). Hızlı ağ trafiğini, paket kaybetmeksizin, etkili bir biçimde incelemektedir. Bro, tek noktada, ağ trafiğini, kural tabanlı olarak inceler. Saldırı tespit sistemleri konumlandıkları yer itibari ile kendileri de saldırılara

açıktırlar. Bro'da ilk defa saldırı tespit sisteminin kendi güvenliği de dikkate alınmış ve sistemin kendini koruması için çeşitli mekanizmalar geliştirilmiştir.

2.3.7 Snort

1998 yılında Martin Roesch tarafından geliştirilmeye başlanmıştır. Şu anda en son 2.6.1.3 sürümü bulunmaktadır [3]. Snort açık kaynak kodlu, yalın ve hızlı bir ağ saldırı tespit sistemidir. Snort, saldırı tespitine ek olarak; trafik analizi, paket günlüğü tutma ve saldırı önleme (intrusion prevention) kabiliyetlerine de sahiptir. Protokol analizi, içerik arama ve karşılaştırma metotlarını kullanarak farklı atak ve denemeleri (örn. arabellek taşırma, port tarama, CGI atakları, Samba sondaları, işletim sistemi izi arama) tespit edebilmektedir. Durum bilgili protokol analizi ve TCP (Transmission Control Protocol), UDP (User Datagram Protocol) akımlarını yeniden kurma özelliğine sahiptir. Bu sayede birçok atağı oturum seviyesinde analiz eder ve saldırganların Snort'u atlatmasını engeller. Snort'ta esnek bir kural tanımlama dili mevcuttur. Bu dil sayesinde saldırı özellikleri ve tespit sonucu yapılacak eylemler tanımlanabilir [12]. Snort, Windows, Linux ve UNIX sistemlerde çalışabilmektedir.

Snort, modüler bir yapıya sahiptir. Şekil 2.7'de gösterildiği üzere 5 temel yapıdan oluşmaktadır.

- **Paket Yakalayıcı:** Bilgisayar ağından bilgisayara iletilen paketleri yakalar. Bu işlem için libpcap veya winpcap kütüphanesini kullanır. Snort genelde fiziksel ortamdan iletilen tüm trafiği izleyecek şekilde yapılandırılır (promiscuous mode).

- **Çözümleyici:** Yakalanan paketleri içerdiği ağ protokollerine uygun olarak çözümler ve sistemdeki veri yapılarına ekler. Protokol analizi yaparak, paketin içerdiği başlık bilgisinin ait olduğu protokole uygun olup olmadığını inceler.

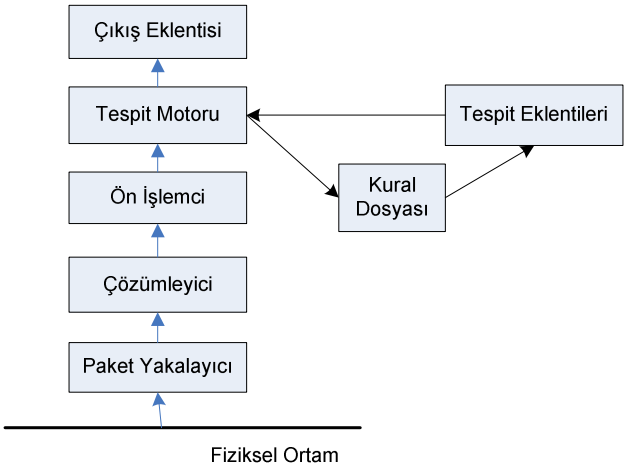
- **Ön İşlemciler:** Ön işlemciler birer süzgeç olarak düşünülebilir. Paketleri analiz aşamasından önce farklı açılardan incelerler. Her ön işlemci bir protokole veya yaklaşıma özgü inceleme yapar. Bu yapı ön işlemciyi yalınlaştırarak hızlı ve verimli çalışmasını sağlar.

- **Kural Dosyaları:** Bilinen saldırılara karşı üretilmiş olan formatlı kural dizileridir. Kural dizileri; protokol, adres ve çıktı eklentileri vb. bilgilerini içerirler. Virüs tanım dosyaları gibi sürekli güncellenirler.

- **Tespit Eklentileri:** Kural dosyalarındaki tanımlara göre örüntü eşleme işlemini gerçekleştirirler.

- **Tespit Motoru:** Tespit eklentilerini kullanarak daha önce belirlenen kuralları gelen paketlere uygular. Değerlendirme sonucu bir saldırı tespit edildiyse sonucu çıkış eklentisine iletir.

- **Çıkış Eklentileri:** Tespit motorunda üretilen değerleri düzenleyerek kullanıcının erişebileceği değişik kaynaklara yönlendirir (Sistem günlükleri, veritabanı, konsol vb.).



Şekil 2.7 Snort'un genel yapısı.

Snort'un işleyişi genel olarak şöyledir. Paket ilk olarak paket yakalayıcı yardımı ile fiziksel ortamdan çekilir. Paketin içerdiği iletişim protokolüne özgü bilgiler belirlenir (ip numarası, kaynak ile hedef port numaraları vb.). Paket, ileriki aşamalarda incelenmek üzere ilgili veri yapısına eklenir. Ayrıca protokol analizi yapılarak hatalı protokol başlık bilgileri henüz ayrıntılı analiz aşamasına gelinmeden tespit edilir. Paket daha sonra sıra ile tüm ön işlemcilerden sırayla geçirilir. Ön işlemden geçen paketlere tespit motoru tarafından kural dosyasında bulunan kurallar uygulanır. Kurallar, ilgili tespit eklentileri yardımı ile paket ile

karşılaştırılır. Bu karşılaştırmalar sonucunda tespit motoru paket hakkında saldırı bilgisi içerip içermediğine dair bir karar verir. Sonuçta üretilecek bir alarm var ise, belirtilen çıkış alarmını oluşturmak üzere gerekli çıkış işleyicilerden geçirilir.

2.3.7.1 Spade

Snort için yazılmış bir ön işlemci eklentisidir. İstatiksel anormallik tespitini kullanmaktadır. Anormallik değeri basit olasılık hesabı ile belirlenir. Tespit için yalnızca veri paketinin başlık bilgisine bakılır. Paketin başlık bilgisindeki belirli değerler incelenerek tekrar sayısı veya miktarına göre istatistiksel bir değer oluşturulur. Daha sonra hesaplanan bu değerlerin tanımlanmış eşik değerlerini aşıp aşmadığı kontrol edilir.

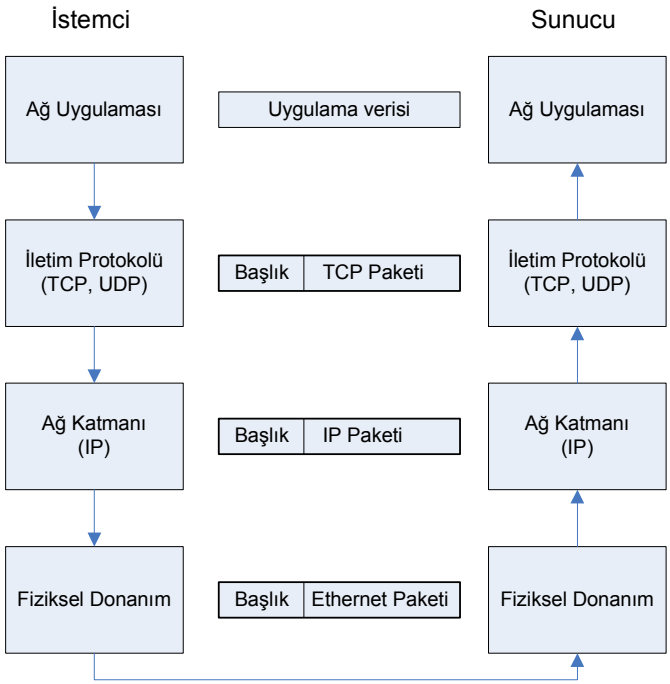
2.4 Yetkisiz Faaliyetler

Internet tasarım olarak zayıf bir güvenlik seviyesine sahiptir. Internet'in kurulduğu yıllarda güvenlik unsuru pek dikkate alınmamıştır. Internet günden güne yaygınlaştıkça üzerinde barındırdığı bilgi ve hizmet miktarı artmaktadır. Kötü niyetli kişiler bu bilgi ve hizmetleri ele geçirmek veya kullanırlılığını engellemek amacıyla Internet üzerinde yetkisiz faaliyetlerde bulunmaktadırlar. Internet'in temel iletişim protokolü TCP/IP'dir. Gerek TCP/IP protokolündeki gerekse de diğer protokol ve uygulamalardaki (UDP, ARP, DNS, SNMP, ICMP, FTP, HTTP, finger v.b.) zayıflıklar bilgisayar ağı saldırılarının temel dayanak noktasını oluşturur. Bu bölümde protokol ve uygulamalardaki zafiyetler ve saldırganlar tarafından kullanılan teknikler incelenmektedir.

2.4.1 TCP/IP

Internet'in temel protokolleri TCP ve IP'dir. TCP/IP Şekil 2.8'de gösterildiği gibi katmanlı bir yapıya sahiptir. Fiziksel katmanda kullanılan Ethernet protokolü datagramların yerel bilgisayar ağı içerisinde iletimini sağlar. Ağ katmanındaki Internet protokolü (IP) yerel bilgisayar ağları arasında paketlerin yönlendirilmesinden ve iletiminden sorumludur. IP protokolü güvenliksiz bir protokoldür. Bunun nedeni veri iletiminin daha hızlı yapılabilmesidir. Eksik olan güvenlik özelliğini ise taşıma katmanındaki iletişim kontrol protokolüne (TCP) bırakır. TCP protokolü bir paketin Internet üzerindeki iki bilgisayar

arasında düzgün ve doğru sırada iletildiğini kontrol eder. Internet üzerinden iletilmek istenen bilgi sırası ile Şekil 2.8'de gösterilen katmanlardan yukarıdan aşağıya iner. Her katmanda o katmana ait başlık bilgisi eklenir. En son katmanda Ethernet başlığı da eklenerek fiziksel ortama verilir. Paketi alan bilgisayar bu sefer aşağıdan yukarıya doğru her katmanda o katmana ait başlık bilgisini açıp yorumlayarak en üst katmana kadar ilerler.



Şekil 2.8 TCP/IP ağındaki veri akışı.

2.4.2 Bilgisayar Ağı Protokollerindeki Suistimaller

ARP (Address Resolution Protocol), IP (Internet Protocol), UDP (User Datagram Protocol), TCP (Transmission Control Protocol) ve ICMP (Internet Control Message Protocol) protokollerindeki aksaklıklar ve bu protokoller kullanılarak yapılan bazı saldırı çeşitleri bu bölümde açıklanmıştır.

2.4.2.1 ARP Aksaklıkları

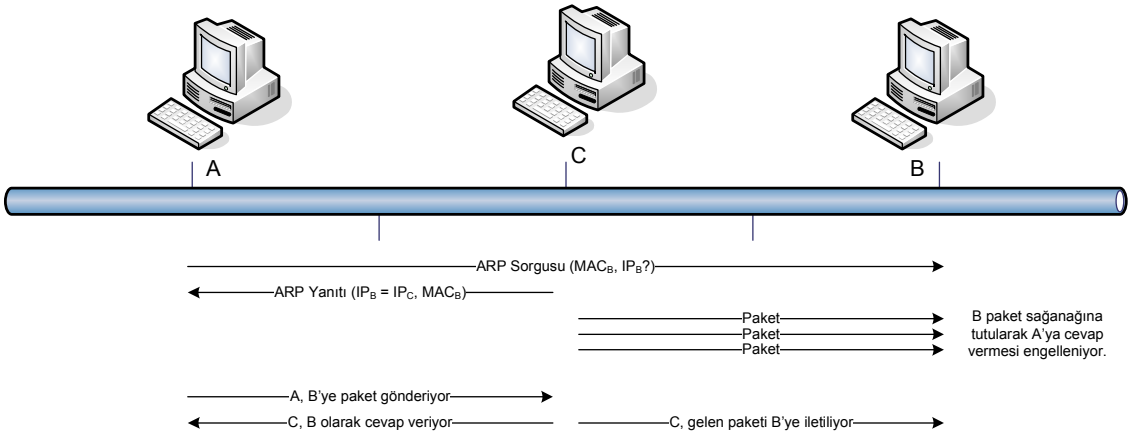
Bilgisayar ağına bağlanan her cihazın benzersiz ve tek bir MAC (Medium Access Protocol) adresi mevcuttur. MAC adresi veri bağı katmanında (data link layer) adresleme yapar. MAC adresi 48 bit uzuğunda olup Ethernet kartına fiziksel olarak yazılmıştır. Bir Ethernet kartı yerel bilgisayar ağındaki başka bir Ethernet kartına veri gönderirken MAC adreslemesini (fiziksel adresleme) kullanır. Yerel bilgisayar ağları arasındaki veri iletişimi için ise 32 bitlik mantıksal IP adreslemesi kullanılır. IP adresi bilinen bir sunucuya veri göndermek için o sunucunun MAC adresinin de bilinmesine gerek vardır. Adres çözümleme protokolü (ARP) IP adresi üzerinden fiziksel adresin tespitini gerçekleştirir. Bilgisayarlar ARP sorgularını, daha verimli olması amacıyla, ARP önbelleklerinde saklarlar. ARP istek ve cevaplarında kimlik doğrulama yapılmaması önemli bir zafiyet olmasına rağmen ARP paketlerinin yerel bilgisayar ağında (LAN) sınırlı kalmasını sağlar. Ancak, yerel bilgisayar ağına erişebilen saldırganlar bu protokolü kullanarak ARP sağanağı, MAC kandırma ve ARP kandırma saldırılarını yapabilirler.

ARP Sağanağı (ARP Flooding): Ethernet altyapısında iki farklı teknoloji kullanılır. Anahtarlamasız ağlarda kullanılan göbek (hub) cihazı, gelen trafiği geldiği bağlantı noktası dışındaki tüm bağlantı noktalarına yollar. Bu durumda gönderilen paket aynı ağ üzerindeki tüm sistemlere de iletilmiş olur. Göbek, ayrıca oluşabilecek çarpışmaları (collision) önlemeye çalışan algoritmalar kullanır. Anahtarlamalı ağlarda ise anahtarlama (switch) cihazı, paketi alıcı bilgisayarın bağlı olduğu bağlantı noktasına iletir. Bu iletimi, bilgisayar ağındaki bilgisayarlar tarafından üretilen ARP sorgularını yakalayarak oluşturduğu ARP tablosu yardımı ile gerçekleştirir. ARP tablosu yerel bilgisayar ağındaki bilgisayarların MAC ile IP adreslerinin ve bağlı oldukları arayüzün tutulduğu bir karşılaştırma tablosudur. Eğer, ARP tablosunda olmayan bir MAC adres ile karşılaşılırsa anahtarlama cihazı göbek cihazı gibi davranarak gelen paketleri kaynak bağlantı noktası dışındaki tüm bağlantı noktalarına iletir. ARP tablosunun boyutu sınırlıdır. ARP tablosu dolduğunda ve yeni bir MAC adresi geldiğinde anahtarlama cihazı iki farklı yöntemden birisini kullanır. Birinci yöntemde anahtarlama cihazı kendi ARP tablosunda bulunmayan fiziksel adresleri, göbek cihazı gibi davranarak, diğer bağlantı noktalarına çoğaltarak gönderir. Burada başarım düşer ancak iletişimin devamlılığı sağlanmış olur. İkinci yöntemde ise tüm ARP tablosu silinerek yeni gelen ARP isteklerine göre tekrar oluşturulur. Ancak, bu yöntemde ARP tablosu oluşturuluncaya dek paketler diğer tüm bağlantı noktalarına da iletilir. Her iki yöntemde de

sınırlı bir süre için de olsa veri trafiği anahtarlama cihazı üzerindeki tüm bilgisayarlar tarafından görülebilir hale gelmektedir. Kötü niyetli bir kişi sürekli olarak farklı MAC adresli paketler üretip, anahtarlama cihazındaki ARP tablosunu şişirerek anahtarlama cihazının göbek cihazı yöntemini kullanmaya başlamasını sağlayabilir. Böylece bilgisayar ağındaki tüm gelen ve giden paketleri alacak şekilde yapılandırılmış (promiscuous mode) bir Ethernet kartı ile bilgisayar ağı üzerinden iletilen şifre, gizli veya hassas bilgiye erişebilir.

MAC Kandırma (MAC Spoofing): Her Ethernet kartında bulunan MAC adresi, o Ethernet kartının üreticisi tarafından konulan tek ve benzersiz bir adrestir. Ancak, her ne kadar MAC adresinin tek ve benzersiz olduğu kabul edilse de Ethernet kartı fiziksel düzeyde tekrar programlanarak MAC adresi değiştirilebilir. Saldırgan kişi kendi bilgisayarındaki MAC adresini yerel bilgisayar ağındaki başka bir bilgisayarın MAC adresi ile aynı olacak şekilde değiştirirse her iki bilgisayarın da yerel ağ bağlantısı kesilecektir. Eğer saldırgan fiziksel adresini varsayılan bilgisayar ağ geçidine (default gateway) ait fiziksel adresi olarak programlarsa, o yerel ağdaki tüm bilgisayarların diğer bilgisayar ağları ile olan bağlantısı kesilmiş olur. Çünkü yerel ağdaki tüm bilgisayarlar dış bilgisayar ağına gönderecekleri paketleri saldırgana ait bilgisayara iletmiş olurlar.

ARP Kandırma (ARP Spoofing): Şekil 2.9'da gösterildiği üzere A bilgisayarı B bilgisayarı ile iletişime geçmek için öncelikle B bilgisayarının IP adresini öğrenmelidir. B bilgisayarının IP adresini öğrenmek için bilgisayar ağına bir ARP paketi gönderir. Bu esnada araya giren kötü niyetli C bilgisayarı ARP isteğine yanıt verir. Aynı anda B bilgisayarının cevap vermesini önlemek için B bilgisayarını paket sağanağına tutar. C bilgisayarı sahte ARP cevabında IP adresi olarak kendi adresini, MAC adresi olarak ise B bilgisayarının MAC adresini yazar. Bu durumda A, C yerine B ile iletişime geçer. C kendine gelen paketi A'dan geliyormuşçasına B'ye ileterek iletişim bağlantısını tamamlar. Artık C, A ile B arasındaki trafiği izler durumdadır.



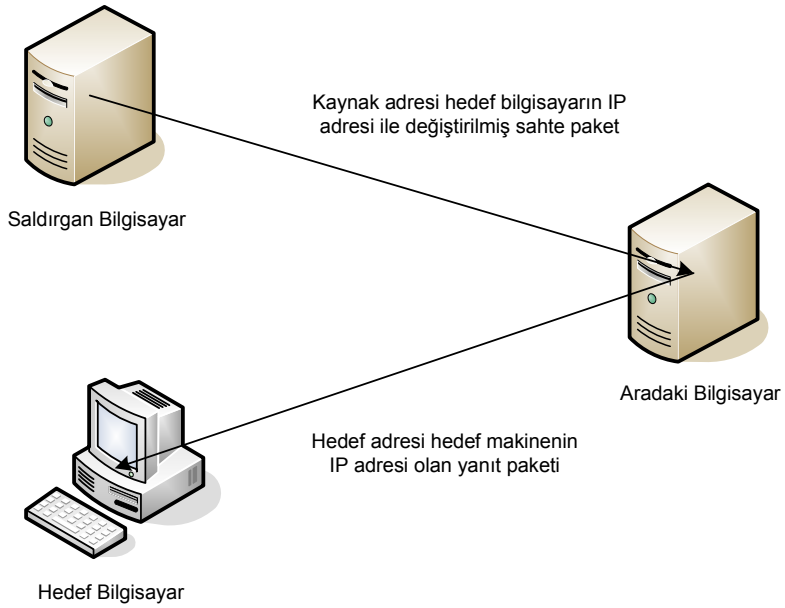
Şekil 2.9 ARP kandırma saldırısı.

2.4.2.2 IP Aksaklıkları

IP daha üst seviyedeki protokolleri taşıyan güvenliksiz bir iletişim protokolüdür. IP datagramların iletimini ve hedefe teslimini sağlar. Ancak, bu iletimde veriyi gönderen kaynağın doğruluğunu kontrol etmez. Bu durum sahte paketlerin üretilmesine veya gerçek paketlerin değiştirilmesine sebep olur.

IP Adresi Kandırma (IP Spoofing): Kusursuz bir tasarımda tüm Internet servis sağlayıcıları gelen (ingress) ve giden (egress) trafiği filtrelemeleri gerekmektedir. Gelen bir paketin hedef IP adresi, yönlendirici cihazın (router) iç bacağındaki bilgisayar ağında tanımlı olmalıdır. Aynı şekilde dışarı çıkan bir paketin kaynak adresi paketin geldiği yerel bilgisayar ağında (LAN) tanımlı olmalıdır. Ayrıca, sadece yerel bilgisayar ağlarında kullanılmak üzere ayrılmış olan özel IP adres aralıklarını [4] kullanarak yerel ağdan Internet'e çıkmaya çalışan trafik engellenmelidir. Ne yazık ki bu kısıtlamalar kenar yönlendirici cihazlarında başarımın gözetilmesi nedeniyle uygulanmamaktadır. Bu da aslında geniş alan bilgisayar ağlarında (WAN) bulunmaması gereken IP adresleri ile karşılaşmamıza neden olmaktadır. Normalde bir bilgisayar gelen paketin kaynağını doğrulama imkânına sahip değildir. İki yönlü iletişim kurulduğunda ise veri iletişimi güvenli bir düzeye gelir. Ancak, iki bilgisayarın iletişim yolundaki bir atlama noktası kullanılarak rahatlıkla bu iletişime karşı saldırı düzenlenebilir. IP adresi kandırma saldırısı, bir bilgisayarın kendi IP adresi dışında başka bir IP adresi kullanarak bilgisayar ağına sahte paketler göndermesidir. Bu saldırı türünün kullanılma nedeni saldırgan kişinin kendisini bu şekilde gizleyebilmesidir. Bu saldırının saldırgan

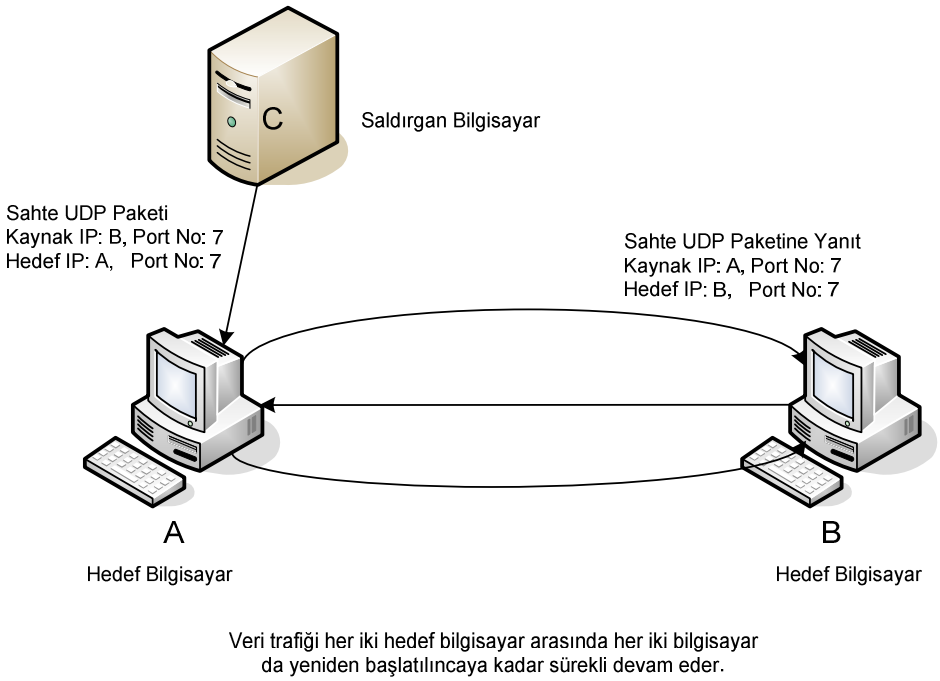
yönünden bir dezavantajı, gönderdiği sahte pakete verilen cevabın kendisine geri dönmeyişidir. Bu yüzden bu saldırılar "gönder ve unut" saldırıları olarak da bilinir. Ancak, bu mekanizma IP adresi kullanılan bilgisayara karşı saldırı olarak da kullanılabilir veya daha önce ele geçirilmiş bir makinenin IP adresi kullanılarak geri dönüşler de elde edilebilir.



Şekil 2.10 IP kandırma saldırısı

TCP protokolünde her gönderilen paketin bağlantı esnasında oluşturulan ve belli bir algoritmaya göre artan bir sıra numarası vardır. Sahte SYN paketine verilen SYN/ACK cevabı hedef makineye ulaştığında daha önce bu pakete ait bir bağlantı bulamayacağından paketin kaynağı olarak gördüğü aradaki bilgisayara RST paketi göndererek bağlantıyı sıfırlar. Bu örnekte de görülebileceği gibi bir sahte paket sonucu iki paket üretilmiştir. Bu da trafik şişirme saldırısına olanak tanımaktadır. UDP protokolünde sıra numarası bulunmamaktadır. Bu da saldırganların işini kolaylaştırmaktadır.

Şekil 2.11'de görüldüğü gibi saldırgan oluşturduğu sahte paketin kaynak ve hedef port adreslerine echo [5] veya chargen [6] servislerinin adresini yazarak diğer iki bilgisayarın karşılıklı olarak birbirlerine sürekli paket göndermelerini sağlayabilir (*fraggle attack*).



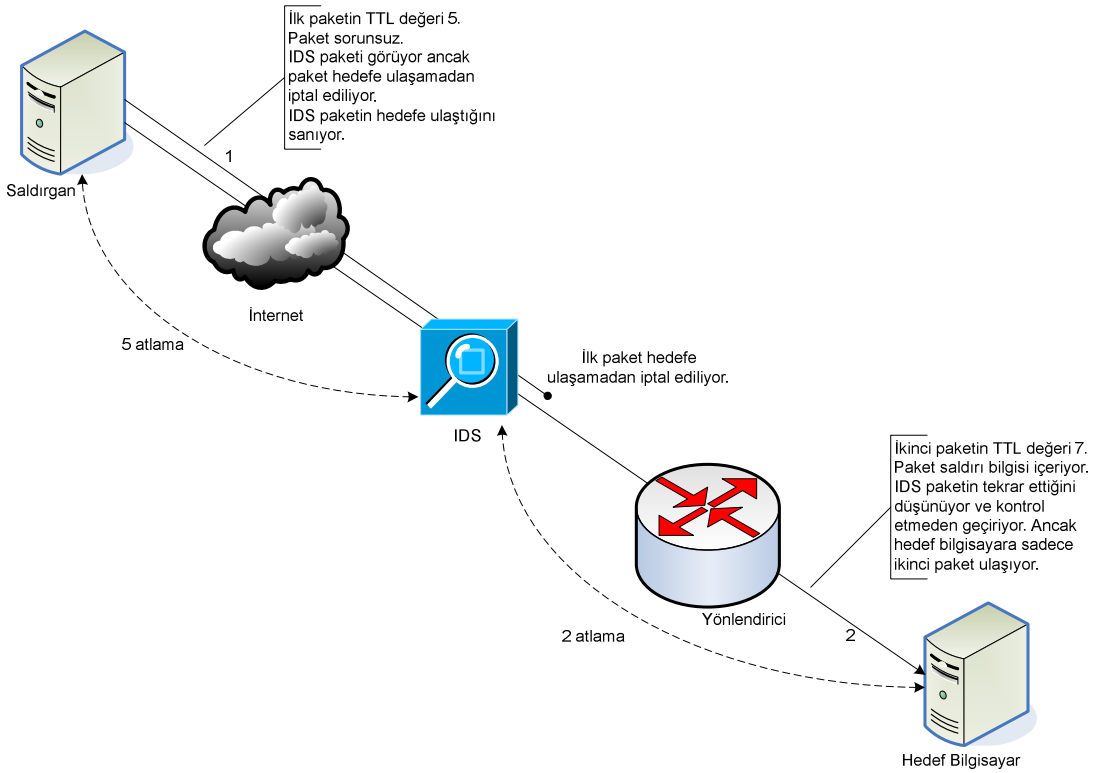
Şekil 2.11 UDP Kandırma Saldırısı

Bir başka benzer atak ise sahte paketin kaynak ve hedef adreslerine saldırılmak istenen bilgisayarın IP ve port adreslerini yazarak ağ trafiğini kendi içinde kısır döngüye dönüştürmek ve bilgisayarın sistem kaynaklarını tüketmektir.

Trafik Şişirme (Traffic Amplification): IP adresi kandırma saldırılarındaki sahte paketlerde tüme gönderim (broadcast) adresleri kullanılarak trafik şişirme yapılabilir. Tümegönderim adresleri, bir altağdaki (subnet) en yüksek IP adresi olup tümegönderim adresi ile gönderilen bir istek o altağdaki her bilgisayara iletilir. TCP bağlantısı noktadan noktaya bir bağlantı olduğu için Tümegönderim kullanılan durumlarda TCP bağlantısı kullanılamaz. UDP ve ICMP gibi bağlantısız olan bu iletişimde saldırgan kişi hedef adresi olarak Tümegönderim adresini ve kaynak adresi olarak da hedef bilgisayarın adresini yazıp paketi bilgisayar ağına verdiğinde (örn: ICMP echo), bu isteğe o yerel ağdaki tüm bilgisayarlar cevap verecektir. Hedef bilgisayara aynı anda çok fazla paket gönderileceğinden hedef bilgisayar işlemez hale gelecektir. Eğer kaynak adresi de bir tümegönderim adresi olarak belirlenirse bu kez istekler her iki altağın tüm bilgisayarları tarafından yanıtlanacağından birbirlerinin ağ trafiğini bloke etmiş olacaklardır. Bu tür saldırıları önlemenin en kolay yolu kenar yönlendiricilerinin tümegönderim adreslerini yönlendirmemeleri yönünde yapılandırmaktır.

IP Paketi Parçalama (IP Packet Fragmentation): Paket temelli bilgisayar ağlarında taşınabilecek en büyük boyutlu paketi belirten maksimum iletim birimi (MTU) bulunur. IP protokolü ile iletilen verinin boyutu MTU değerinden eğer büyük ise parçalara ayrılarak taşınır. Verinin parçalanarak taşınması işlemi kötü niyetli kişiler tarafından suistimal edilebilir. Özellikle farklı durum ve birleşimlerde gelen parçalanmış verinin nasıl birleştirileceğine dair algoritmik yaklaşımların açıklarından faydalanılır. Örneğin, farklı içerikteki ancak aynı sıra numaralı iki parçalı paket gönderildiğinde iletiyi alan bilgisayar hangi paketi kullanacağına karar vermek zorundadır. İki parçalı paketin içeriği birbirlerine örtüşecek şekilde yollandığında iletiyi alan bilgisayar bu durum ile başa çıkmak zorundadır. IP başlığında bulunan "*fragment offset*" değeri ile oynanarak bir ip paketinin boyutu olması gereken en büyük değer olan 65.535 byte'tan fazla gösterilebilir. Bu durumda eğer iletiyi alan bilgisayar gelen her paket için arabelleğinde açtığı alan en fazla 65.535 byte ise arabellek taşırma saldırısına maruz kalabilir. "ICMP echo" paketleri ile yapılan bu saldırıya "*ping of death*" saldırısı adı verilir. Son paket hariç, gönderilen IP parçaları sekizin katları olacak şekilde boyutlandırılır. Eğer gönderilen parçanın boyutu sekizin katı değilse bu durum kontrol edilmelidir. Bazı güvenlik duvarı yazılımları parçalı olarak gönderilen paketlerden ilk gelen paketi kontrol ederek kural kontrolü yapmakta ve diğer parçaları ilk pakete göre değerlendirmektedir. Bu durumda saldırgan ilk paketin bilgilerini güvenlik duvarını geçebilecek şekilde yapılandırarak diğer paketleri ise normalde erişimi kısıtlanmış kaynaklara göre yapılandırabilir [7].

TTL Sorunları: TTL (Time to Live), Internet ağında paketlerin sonsuza değin dolanıp durmalarını ve dolayısıyla protokol hatalarına yol açmalarını engellemek üzere saniye ya da hoplama sayısı olarak biçilen ömürdür (Bülent Sankur, 2005). Şekil 2.12'te görüldüğü üzere TTL değeri kullanılarak saldırı tespit sistemlerini yanıltmak mümkün olabilmektedir. Hedefe gönderilen ilk paketin TTL değeri IDS'i geçecek kadar büyük ancak hedefe ulaşamayacak kadar küçük bir değer seçilir. Bu ilk paketin içeriği sorunsuz ve doğru olduğu için IDS tarafından kabul edilir. Ancak hedefe ulaşmadığı için bir hata veya alarm durumu da oluşmaz. İkinci paket ise ilk paketin tekrarı olarak yapılandırılır. Ancak, ikinci paketin içeriğinde kötü niyetli veri bulunmaktadır ve TTL değeri hedef bilgisayara ulaşacak şekilde yapılandırılmıştır. Paketi alan IDS, ilk paketin tekrarı olarak gördüğü ikinci paketi de kabul eder. İkinci paket bu defa hedefe ulaşır ve saldırıyı gerçekleştirir.



Şekil 2.12 TTL değeri ile oynayarak IDS sisteminin kandırılması.

2.4.2.3 UDP Aksaklıkları

UDP protokolü bağlantısız bir protokol olduğundan ve kaynak doğrulaması yapmadığından dolayı bilgisayar ağı saldırılarına zemin hazırlayan bir protokoldür. Bunun dışında protokolün başlık kısmında bulunan sağlama toplamı değerinin isteğe bağlı olması bir başka zayıflıktır. Eğer sağlama toplama değeri 0 ise paketin sağlaması yapılmamaktadır. Bu durum saldırganların kolaylıkla UDP paketlerini değiştirme, ekleme veya silme yapabilmelerine olanak tanımaktadır. Günümüz bilgisayarlarında hem iletişim hem de işlemci başarımlarının sağlama işlemi için yeterli olduğu düşünüldüğünde bu özellikteki paketlere karşı ihtiyatlı olmak gerekmektedir.

2.4.2.4 TCP Aksaklıkları

TCP bağlantılı bir protokol olduğundan ve noktadan noktaya kaynak doğrulaması, bütünlük kontrolü yapıldığı için IP'ye göre daha güvenli bir protokoldür. Eğer bir TCP bağlantısının

izlediği yol üzerindeki bir yönlendirici üzerinde bulunulmuyorsa bu bağlantıyı kötüye kullanmak çok zordur. Buna rağmen bazı protokolden kaynaklanan aksaklıkları bulunmaktadır.

TCP Yeniden İletim: TCP, IP protokolünün güvensiz yapısını güvenlikli bir hale getirmek için öncelikli olarak yeniden iletim mekanizmasından faydalanır. Tekrar iletilen bir paketin içeriği ilk paket ile birebir aynı olmalıdır. Eğer aynı değilse IDS tarafından engellenmelidir.

TCP Bayrakları (TCP Flags): TCP başlığında iletişimin durumunu belirten bayrak bilgileri bulunur. Bu bayraklar farklı birleşimlerde olabilirken bazı birleşimlerde bulunmaları normal değildir ve bir saldırı habercisi olabilir. Çizelge 2.6'da normal bir iletişiminde aynı anda bulumamaları gereken TCP durum bayrakları gösterilmiştir.

Çizelge 2.6 TCP durum bayrakları

TCP Bayrakları	**Sorun**
Hiçbiri	Hiçbir bayrak değeri olmayan paket geçerli bir TCP işlemi değildir. TCP başlığında en azından ya SYN, ACK ya da FIN/RST bayrakları olmalıdır.
SYN/FIN	SYN oturum açma FIN oturum sonlandırma bayrağıdır ve her ikisi aynı anda uygulanamaz.
SYN/RST	SYN oturum açma RST oturum sonlandırma bayrağıdır ve her ikisi aynı anda uygulanamaz.
SYN/FIN/ACK	SYN oturum açma, ACK oturumu devam ettirme, FIN oturum sonlandırma bayrağıdır ve her üçü aynı anda uygulanamaz.
SYN/RST/ACK	SYN oturum açma, ACK oturumu devam ettirme, RST oturum sonlandırma bayrağıdır ve her üçü aynı anda uygulanamaz.
Tümü	Tüm bayraklar aynı anda uygulanamaz.

SYN Sağanağı (SYN Floods): Çoğu TCP uygulamasının, SYN sağanağı olarak da bilinen, kaynak tüketme saldırına karşı zafiyeti vardır. TCP oturumu oluşturmak için öncelikle bir SYN paketi gönderilir. Sunucu ise bu yeni oturum için arabelleğinde yer açar ve isteğe SYN/ACK paketi ile cevap verir. Bu anda TCP oturumu tam olarak kurulmuş değildir.

Sunucu üç yönlü el sıkışmanın tamamlanması için istemciden son bir ACK paketi beklemektedir. Eğer art arda yeni TCP oturumları açmak üzere SYN paketleri gönderilirse sunucu her yeni oturum için fiziksel olarak kısıtlı olan arabelleğinden bir yer ayıracaktır. Ancak bir süre sonra arabellek dolacağından hiçbir bağlantı isteğine cevap veremeyecektir. Her bağlantının bir zaman aşımı süresi vardır. Ancak bu süre genellikle arabelleği taşırmak için yeterli olmaktadır. Bu sorunun üstesinden gelebilmek için farklı yöntemler izlenmektedir. İlk yöntem aynı anda açılan TCP oturumu sayısını kısıtlamaktır. Bu durumda TCP oturumu sayısı belirli bir limite ulaştığında sunucu tarafından reddedileceklerdir. Ancak bu yöntemin dezavantajı iptal edilen paketlerin, TCP tasarımında olduğu üzere, istemci tarafından tekrar gönderilmesidir. Ayrıca bilgisayara sızılmamış olsa da bir bakıma hizmet aksatılmış olmaktadır. İkinci yöntem ise yeni bir oturum isteği geldiğinde arabellekte bekleyen ve henüz tam bağlantı sağlanamamış oturumlardan birini silmek ve yeni oturuma yer açmaktır. Eski oturum uzun zamandır tamamlanamadığından saldırı amaçlı yaratılmış olabilir. Ancak yüksek hızlı bağlantılarda geçerli oturumun da silinme ihtimali vardır. En sık kullanılan üçüncü yöntem ise SYN kurabiyeleri kullanmaktır. Sunucu SYN/ACK bilgisi ile birlikte, gelen istekten ürettiği şifreli sıra numarasını gönderir. Ancak arabellekte bu oturum ile ilgili bir yer ayırmaz. İstemciden ACK paketi geldiğinde sunucu tekrar şifreli bir bilgi üretir ve gönderdiği veri ile kıyaslar. Eğer her iki değer de aynı ise geçerli bir TCP oturumu açılır.

Sıra Numarasını Tahmin Etmek: Her TCP oturumunun rasgele belirlenen bir sıra numarası bulunur. Eğer bu sıra numarası bilinirse iletişim halindeki bilgisayarlar kandırılabilir. Oturuma aktif olarak müdahale edilebilir. Günümüzdeki bilgisayarlarda sıra numarasını tahmini zor bir şekilde belirlenmektedir. Ancak, donanımsal uygulamalarda başarım göz önünde bulundurularak sıra numarası sıralı verilebilir.

2.4.2.5 ICMP Aksaklıkları

ICMP bağlantısız bir protokol olduğundan saldırılara uygun bir zemin hasırlamaktadır. ICMP protokolü kullanılarak genellikle hizmet aksatma saldırıları yapılmaktadır.

İlk saldırı yöntemi, sunucuya erişmek isteyen istemcilere “Sunucu erişilemez” mesajı içeren ICMP mesajları yollamaktır. Bu durumda istemciler TCP oturumlarını kapatacaklar ve sunucudan hizmet alamayacaklardır.

İkinci yöntem, geçerli bir TCP oturumundaki istemci veya sunucuya “düzeltme” mesajı içeren ICMP paketi göndererek iletişim hızını düşürmek ve hizmeti aksatmaktır.

Tam olarak ICMP ile ilgili olmasa da gizli iletişim kanallarının kullanımında ICMP sıklıkla kullanılır. Ele geçirilen bir bilgisayar ile konuşmak isteyen saldırgan kendi oluşturduğu gizli iletişim kanallarını kullanır. İletişim kanalı olarak ICMP paketlerinde kullanılmayan alanlardan faydalanılabilir. ICMP başlığı ufak olduğu için geri kalan alan bu iş için kullanılabilir. Bu kanalların IDS tarafından tespit edilebilmesini güçleştirmek için veri trafiği seyrekleştirilir. Böylece IDS belirli bir aksaklık tespit etmesi güçleştirilir. Ayrıca trafik PGP veya SSL kullanılarak şifrelenir ve içeriğinin tespiti önlenir.

2.4.3 Programlardaki ve Protokollerdeki Suistimaller

Günümüzde sıklıkla kullanılan bazı program ve protokoller Internet’in yaygınlaşmaya başladığı ilk yıllarda güvenlik unsuru önemsenmeden tasarlandı. Büyük çoğunluğu kullanıcı adı, şifre ve veri iletişimini düz metin olarak yapmaktadır. Her ne kadar bugün kullanılan anahtarlama alt yapısı araya girmeleri zorlaştırsa da araya girip hassas veriye erişmek imkânsız değildir. Günümüzde bu program ve protokollerin güvenli muadilleri gerçeklenmiş olsa da kullanım oranı oldukça düşüktür. Güvenli protokol ve programlar verinin güvenliğini ve bütünlüğünü sağlamak için şifreleme algoritmalarından faydalanırlar. Her iki bilgisayar arasında iletilen veri şifrelenerek gönderilir. Bu durum saldırı tespit sistemlerinin bilgisayar ağı üzerinde akan şifreli trafiği analiz edememesine neden olur. Bu sorunun çözümü için ya saldırı tespit sistemine deşifre işlemi yapan bir parça eklemek ya da bilgisayar şifreli metini deşifre ettikten sonra analiz yapmak gerekmektedir. Bazı saldırı yöntemleri ise Internet üzerinde iletilen veri ile ilgilenmek yerine direkt sistem üzerindeki servis ve programları hedef alır. Saldırgan, program ve servislerin tasarımdan veya yanlış yapılandırılmasından kaynaklanan zafiyetleri kullanır.

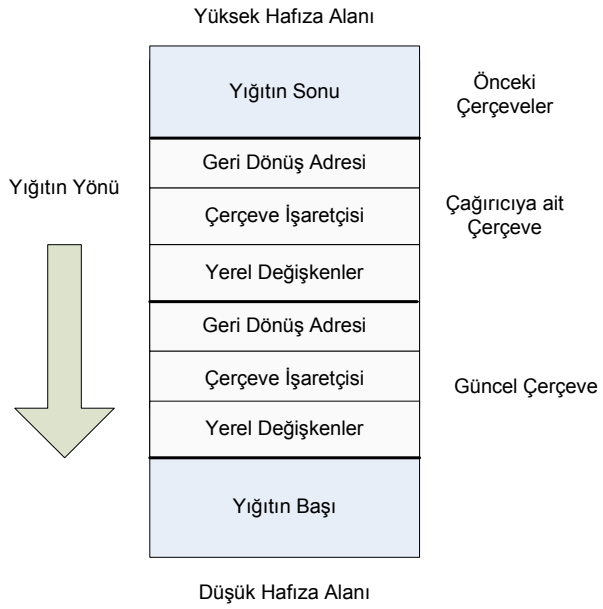
2.4.3.1 Arabellek Taşırma Saldırısı

Uzaktan yapılan güvenlik ihlallerinin büyük bir bölümü arabelleklerin çeşitli yollardan taşırılması ile gerçekleştirilir. Arabellek, bilgisayar hafızasında bir değişken, program veya servis için ayrılmış alandır. Arabellek taşırma zayıflıkları işletim sisteminin hafıza ve erişim

güvenliği mekanizmalarındaki aksaklıklardan meydana gelebileceği gibi bazı programlama dillerinin (Örn. C, C++) tasarımından da kaynaklanmaktadır. Bunun dışında yazılımların hatalı ve güvenlik unsuru dikkate alınmadan kodlanması da bu tür saldırılara sebebiyet vermektedir. Tanımlanan arabelleğin sınırlı bir boyutu vardır. Eğer bu boyutun ötesine geçilirse başka bir programın arabelleğine izinsiz erişilmiş olur. Hedef sistemde kötü niyetli bir program çalıştırıldığında arabelleği olması gerekenden fazla yükleyerek taşıracak ve programına gömdüğü kötü niyetli kod parçasını çalıştırarak sisteme erişim hakkı elde edecektir. Aşağıdaki kod parçasında boyutu 10 olan *char* tipinde bir dizi tanımlanmıştır. Ancak, *strcpy* fonksiyonu ile boyutu 10'dan büyük bir katar bu diziye kopyalanmıştır. Bu durumda daha önce ayrılan arabellek taşırılmış olur.

```
main()    {
        char buff[10];
        strcpy(buff, "Bu katar arabelleğin taşmasına neden olur.");
}
```

Arabellekler bilgisayarda genellikle yığıt (stack) yapılarında saklanırlar. Arabellek taşırma saldırılarını anlamak için öncelikle yığıt yapısını incelemek gerekmektedir. Yığıt yapıları işlemci mimarilerine göre farklılık gösterebilir. Bu çalışmada, Şekil 2.13'te gösterilen Intel 80x86 mimarisi temel alınacaktır. Yığıt işaretçisi, "*stack pointer*" (SP) yığıtın hafıza alanındaki başlangıç adresini belirtir. Bir fonksiyon veya altprogram çağrıldığında parametreler, dönüş adresi ve çerçeve işaretçisi *"frame pointer"* (FP) yığıta basılır.



Şekil 2.13 Yığıt yapısı.

Kötü tasarlanmış bir program her zaman yığıt taşmasına sebep olabilir. Ancak bu zayıflıktan faydalanarak sisteme saldırmak daha fazla bilgi birikimi ve dikkat gerektirir. Programa verilen girdi özel olarak oluşturulur. Girdi içerisinde sisteme erişimi sağlayacak kod parçası bulunur. Bu kod parçası genellikle bir kabuk (shell) oturumu açar. Bu kod parçasının başlangıç adresi, yığıta basılmış olan program geri dönüş adresinin üzerine yazılır. Geri dönüş adresinin yeri ve taşırmanın ne kadar olacağı tam olarak bilinmelidir. Saldırganlar taşırma boyutunu NOP komutu ile düzenlerler. NOP boş bir komuttur ve sistem durumunda bir değişikliğe yol açmaz. Bu komut, geri dönüş adresinin yığıttaki yerinin tam olarak bilinemediği durumlarda saldırgana yeri tahmini olarak bulma imkanı verir. Yazılan taşırma kodunda "0" veya satır sonu karakterlerinin bulunmaması gerekir. Bu karakterler katar yapısı için özel anlam ifade ederler ve katarı sonlandırırlar. Saldırganlar bu sorunu farklı komut birleşim kullanarak çözerler. Örneğin "0" kullanmak yerine XOR komutu kullanılarak 0 değeri elde edilebilir. Saldırı tespit sistemleri sisteme verilen katar bilgisine bakarak bu bilginin zararlı olup olmadığını analiz edebilirler. Örneğin bir katarda çok fazla ve ardı ardına NOP (16'lık: 0x90) komutunun bulunması bir saldırıya işarettir. Saldırganlar bu tür sistemleri atlatmak için bir veya birden fazla komutun istenilen başka bir komutun işlevselliğini sağlamak için kullanmaktadırlar. Örneğin Intel 80x86 mimarisinde NOP komutunun işlevselliğini taklit edebilecek 55 komut bulunmaktadır. Sistemi taşıran veri içerisinde her zaman saldırı kodu gömülü olmayabilir. Standart sistem kütüphanelerinin bilgisayar

hafızasında bulundukları konum genelde öngörülebilir. Taşırma saldırısında geri dönüş adresi olarak bu kütüphanelerin adresi yazılarak kütüphaneler çalıştırılabilir. C programlama dilinde standart giriş ve çıkış fonksiyonlarında genellikle giriş değişkeni ve format bilgisi olmak üzere iki temel parametre bulunur. Bir başka taşırma saldırı çeşidinde ise giriş değişkeni yerine format bilgisini kullanılır.

Arabellek taşırma saldırılarından kaçınmak için izlenebilecek birkaç yol vardır. Arabellek taşırma saldırılarının ana sebebi hatalı veya eksik yazılmış yazılımlardır. Yazılımda her giriş ekranı dikkatlice incelenmelidir. Giriş fonksiyonun alfa nümerik karakterler dışındaki karakterleri kabul etmemesi ve veri uzunluğunu sınırlandırması gerekmektedir. Bazı programlama dilleri (Örn. C, C++) yapıları gereği taşırma saldırılarına karşı zayıflıkları vardır. Mümkünse daha güvenli programa dillerini kullanmak (Örn. Java) veya mevcut zafiyetli kütüphanelere alternatif olarak yazılan güvenli sürümlerini kullanmak gerekmektedir. Sistem bazında bakıldığında ise kullanılmayan servisleri kapatmak ve kullanılanların ise en son sürümlerini elde etmek bu tip saldırılara karşı etkili yöntemlerdir. Bunun dışında işletim sistemi düzeyinde gerçeklenen bazı mekanizmalar mevcuttur. Sistem kütüphanelerinin hafızadaki konumlarını rasgele belirlemek, yığıttaki geri dönüş adresinin etrafına belirli bir algoritma ile belirlenmiş kontrol değeri yerleştirmek (kanarya değeri) başlıca yöntemlerdir.

2.4.3.2 FTP

FTP, iki bilgisayarın bilgisayar ağı üzerinden dosya transferi gerçekleştirmesi için tasarlanmış bir protokoldür. FTP protokolü RFC 959 [8] standardı ile tanımlanmıştır. FTP veri ve kontrol bilgisini düz metin olarak iletir. Ancak, diğer protokollerden farklı olarak kontrol ve veriyi farklı portlardan gönderir. İletişimin güvenliksiz olmasını bir kenara bırakırsak, FTP sunucularındaki ortak paylaşım alanları, anonim kullanıcılar ve erişim kontrolü saldırıya en açık alanlardır. FTP sunucusunun doğru ve tam yapılandırılması, kullanıcı ve yetki tanımlamalarının kesin ve net olarak yapılması, paylaşım alanlarının okuma veya yazma yönünde olacak şekilde belirlenmesi saldırı riskini en aza indirecektir. Saldırı tespit sistemleri gerek içerik bazında gerekse protokol bazında FTP üzerinden yapılan saldırıları tespit edebilmektedir. Özellikle bazı FTP komutları (PORT ve SITE EXEC) IDS'ler tarafından varsayılan olarak reddedilmektedir. PORT komutu istemcinin istediği bir port numarası ile sunucuya bağlanmasını sağlar. Ele geçirilen bir bilgisayarda çalıştırılan FTP istemcisi port adresi olarak farklı bir servisin port numarasını verebilir ve sunucudan çekeceği kötü niyetli

kodla bu servisi çalışmaz hale getirebilir veya kendisini o servisin yerine koyabilir. SITE EXEC komutu ise sunucu tarafında sistem komutu çalıştırır.

2.4.3.3 SSH

SSH (Secure Shell); Telnet, FTP ve Rlogin gibi düz metin temelli protokollere karşılık şifreleme yapan müdavili olarak geliştirilmiştir. SSH protokolü veriyi ele geçirme veya bütünlüğünü bozmaya yönelik saldırılara karşı şifreleme yöntemini kullanır. Protokolün güvenlikli olmasına karşın klavye tuşlarını izleyen casus araçlar kullanıcı adlarını ve şifreleri henüz protokol şifreleme yapmadan önce ele geçirebilir.

2.4.3.4 DNS

Internet üzerindeki her bilgisayarın tek ve benzersiz bir IP adresi bulunmaktadır. Ancak, bu adres kişiler için hatırlanması çok zor sayısal bir bilgidir. Bunun yerine anlamlı harflerden oluşan adlandırmalar kullanılmaktadır. Ancak, bu anlamlı adreslerin bilgisayarların anlayacağı IP adreslerine dönüşümü gerekmektedir. Bu dönüşümü DNS (Domain Name System) yapmaktadır. DNS, başarım kaygıları nedeni ile genellikle UDP protokolünü kullanır. DNS servisini devre dışı bırakma, sahte DNS sorguları üretme, sahte DNS sunucusu oluşturma, DNS sorgusu sağanağı ile DNS tablolarını değiştirme başlıca DNS saldırılarıdır. DNS saldırılarını önlemek için daha güvenli olan TCP protokolünü kullanmak, kimlik doğrulama yapmak, dış bilgisayar ağlarından gelen sorgulamaları engellemek gerekmektedir.

2.4.3.5 HTTP ve WWW

HTTP (Hypertext Transfer Protocol), WWW (World Wide Web)'in en popüler protokolüdür. 1990'lı yıllarda geliştirilen HTTP ve WWW Internet'in hızla yaygınlaşmasına büyük katkıda bulunmuştur. HTTP, Internet üzerinden metin temelli zengin içerik iletimi amacı ile geliştirilmiştir. HTTP protokolü ilk yıllarda sadece HTML (Hypertext Markup Language) ile yazılan dosyaların transferi için kullanılırken günümüzde değişken içerik sağlayan farklı betik dilleri de desteklenmektedir. HTTP, WWW üzerindeki tüm dosyaları URL (Uniform Resource Locator) etiketleri ile tanımlar.

Internet'te en çok kullanılan protokol olması sebebi ile HTTP protokolüne ve bu protokol üzerinden verilen servislere, web sayfalarına yapılan saldırılar günümüzdeki saldırıların önemli bir kısmını oluşturmaktadır. HTTP sunucu ve istemci olarak iki yönden incelenmelidir. Sunucu tarafında en çok kullanılan [11] sunucu yazılımları Apache [9] ve Microsoft IIS'dir [10]. Bu sunucular birçok özellik barındırmalarına karşın bu özelliklerin birçoğu kullanılmayan ve saldırıya açık özelliklerdir. Gerekli olmayan özelliklerin kapatılması sunucu güvenliği için başlıca önlemdir. Varsayılan web sayfasının bulunmadığı durumlarda dizinin listelenmesi çoğu durumda gereksizdir ve saldırgan kişinin sisteminiz hakkında bilgi sahibi olmasına neden olur. Web sunucularının kısıtlı haklar tanınmış farklı bir kullanıcı ile çalıştırılması gerekmektedir. Unicode, URL veya dizin üzerinden yapılan belirli saldırılara karşı sunucuların yamalandığından emin olunmalıdır. CGI (Common Gateway Interface) WWW 'in ilk yıllarında kullanılan bir dinamik içerik ara yüzüdür. Ancak günümüz güvenlik koşullarına göre çok tehlikeli zafiyetleri bulunmaktadır. CGI özellikle arabellek taşırma ve komut sızdırma saldırılarına karşı zayıftır. Bu nedenle yeni teknolojilere geçmek güvenlik açısından önemlidir. Web sunucularına yapılan hizmet aksatma saldırılarına karşı aynı anda kabul edilecek bağlantı sayısı sınırlandırılmalı ve sistem günlükleri düzenli bir şekilde incelenmelidir. İstemci tarafında web tarayıcıları bulunmaktadır. Web tarayıcılarındaki yazılım hataları bu ürünleri kullanan kullanıcıların saldırıya açık hale getirmektedir. Web tarayıcılarının en son sürümlerini kullanmak ve güncellemeleri uygulamak gerekmektedir. Web tarayıcıları ActiveX veya Java teknolojileri ile istemci tarafında komut çalıştırabilmektedirler. Normalde her iki teknoloji de kullanıcı tarafında erişim ve güvenlik mekanizmaları geliştirmiştir. Ancak, hatalı veya bilinçsiz kullanım sık karşılaşılan bir durumdur. ActiveX programlarının istemci tarafında çalışmasını denetleyen bir sertifika yapısı mevcuttur. Her ActiveX objesinin sertifikası mevcuttur ve kullanıcı bu sertifikaya bakarak programın çalışmasına karar verir. Ne yazık ki çoğu kullanıcı bu kararı verecek yeterli uzmanlık seviyesine sahip değildir. Daha da vahimi bu sertifikalar kolaylıkla kötü amaçlı kullanıcılar tarafından elde edilebilir. Bu tür objelerin yüklenmesinde web tarayıcıları veya işletim sistemi kullanıcıyı uyaran mesajlar gönderir. Ancak, bu uyarılar çoğu kullanıcı tarafından dikkate alınmaz. Bir başka saldırı türü de sosyal mühendisliktir. Web sayfalarının bir kopyasını oluşturarak veya kandırmaca bir e-posta mesajı yardımı ile kullanıcıların hassas bilgilerini ele geçirmek mümkün olmaktadır. Bu tür saldırılara karşı alınacak önlemler teknik olmaktan çok bilgisayar kullanıcılarını güvenlik konusunda bilinçlendirmekten geçmektedir. Günümüzde kullanılan dinamik sayfaları arka tarafta bir

veritabanı sistemi ile desteklenmektedir. SQL (Structured Query Language), veritabanı sorgulama dilidir ve web sayfalarının tasarımında dinamik verileri elde etmek için kullanılır. Web sayfalarında bulunan web formları aracılığı ile alınan giriş verisi ile SQL sorguları oluşturulur ve veritabanından istenen veriler kullanıcıya geri döndürülür. Saldırganlar web formları aracılığı ile saldırı verisini SQL sorguları ile birleştirir ve veritabanında hatalı işleme sebep olurlar. Web formlarına özel karakterler ve SQL parametreleri ekleyerek SQL sorgularına sızarlar. Bu tür saldırıları önlemenin yolları hem istemci hem de sunucu tarafında giriş değerlerini kontrol etmek ayrıca veritabanı ve web sunucusunu bu tür saldırılara karşı yapılandırmaktır.

2.4.3.6 SNMP

SNMP bilgisayar ağ trafiğini yönetmek için kullanılan bir protokoldür. Bilgisayar ağında bulunan cihazların yapılandırılması, izlenmesi ve sistem günlüklerinin toplanması SNMP protokolü aracılığı ile yapılabilmektedir. SNMP iletişimde şifreleme veya erişim kontrolü yapmaz. Güvenlik amacı ile topluluk isimlerini (community string) kullanır. Topluluk isimleri okuma ve hem okuma hem de yazma olmak üzere iki adettir. Aynı topluluk ismine sahip bilgisayar ağ cihazları topluluk isminin türüne göre birbirleriyle bilgi alışverişinde bulunabilirler ve yapılandırılabilirler. SNMP yerel bilgisayar ağlarının yönetimi için kullanıldığından SNMP paketlerinin dış bilgisayar ağlarına iletilmemesi gerekmektedir. Bilgisayar ağ cihazlarında topluluk isimleri varsayılan olarak aynı değerler ile gelmektedir. Saldırganlar tarafından bilinen bu değerler, saldırganlara yapılandırılmamış bilgisayar ağları hakkında kolaylıkla bilgi edinme veya yeniden yapılandırma olanağı sağlar. Bu nedenle tüm bilgisayar ağ cihazlarının SNMP topluluk isimleri varsayılan değerlerinden farklı bir değere ayarlanmalıdır ve sıklıkla değiştirilmelidir.

3. TASARIM

Saldırı tespit sistemleri, anormallik tespiti ve imza tabanlı olmak üzere iki temel yaklaşım içerir. Bu yaklaşımların birbirlerine göre avantaj ve dezavantajları vardır. Gerçeklenmek istenen sistemde her iki yaklaşım da kullanılarak saldırı tespit başarısını arttırmak amaçlanmaktadır. İmza tabanlı sistemlerde, imza veritabanını güncel tutmak çok fazla deneyim ve zaman gerektirmektedir. Profesyonelce hazırlanmış bir sistemde güncel saldırılar sürekli olarak takip edilmeli ve bu saldırılara karşı imza veritabanı güncellenmelidir. İmza tabanlı sistemlerin bu zorluğuna karşı sistem tasarımında Snort yazılımından faydalanılması düşünülmüştür. Snort açık kaynak kodlu olması nedeniyle hem özgürce kullanıma hem de geliştirmeye açık bir sistemdir. Snort, günümüzde en sık kullanılan saldırı tespit sistemi olması nedeni ile imza veritabanı her zaman güncellenmektedir. Snort'un tasarımda kullanılması ile aşağıda belirtilen avantajları kullanılmaktadır.

- Birden çok ağ protokolünü destekleme
- Taşınabilirlik
- Standart bir yapı oluşturma
- Güncel imza veritabanı
- Zengin uyarı çıkış eklentileri

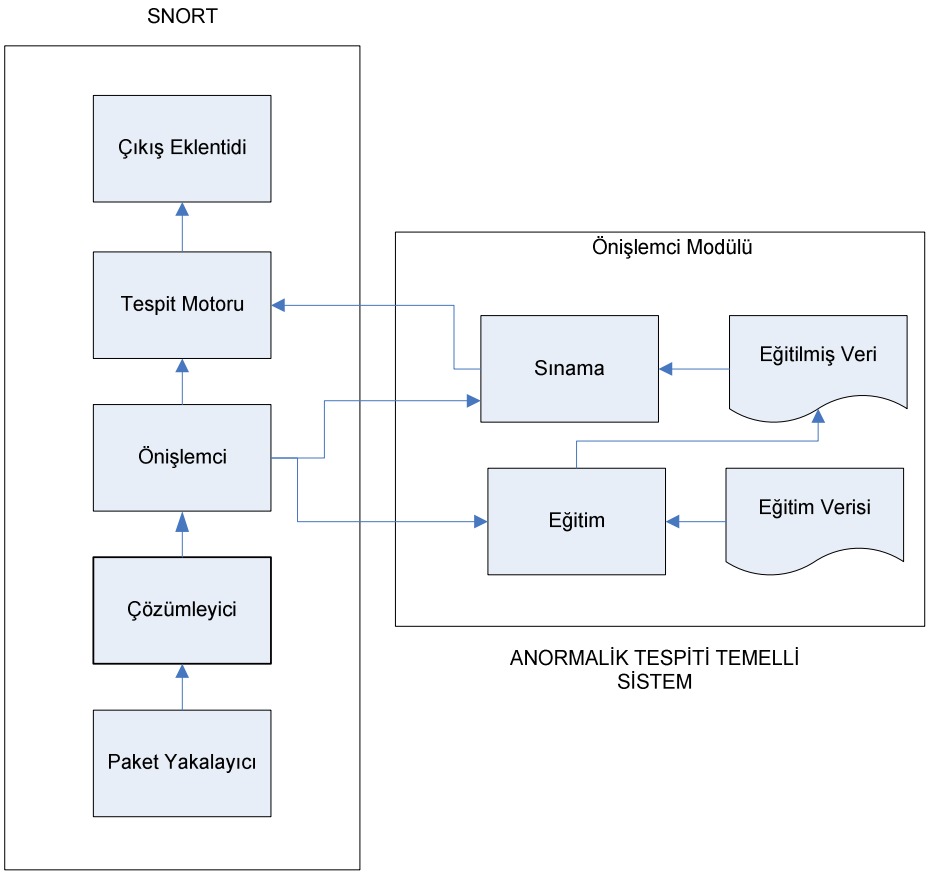
Anormallik tespiti yaklaşımı ile aşağıda belirtilen hedefleri gerçekleştirmek amaçlanmıştır.

- Doğruluk oranını arttırma
- Yanlış-pozitif oranını düşürme
- Bilinmeyen saldırı çeşitlerini tespit edebilme

Anormallik tespitinde temel zorluklar yüksek yanlış pozitif oranı, algoritmaların fazla işlem gücü gerektirmesi ve eğitim kümesinin oluşturulması için çok fazla örneğe ihtiyaç duyulması olarak sıralanabilir (Lee et al., 2001). Bu zorlukları yenmek için sistemin aşağıda belirtilen özelliklerinin yükseltilmesi amaçlanmıştır.

- Doğruluk
- Verimlilik
- Uygulanabilirlik

Şekil 3.1'de görüldüğü üzere; sistem, Snort'a ön işlemci eklentisi olacak şekilde tasarlanmıştır.



Şekil 3.1 Tasarımın genel görünümü.

3.1 Sistemin Çalışması

Sistem, Snort'a ait bir parça olarak tasarlandığı için başlama parametreleri ve sırası Snort üzerinde yapılandırılmaktadır. Sistemin çalışma şeklini belirlemek için aşağıdaki dosyalar yapılandırılmaktadır.

- **Snort.conf:** Snort'un temel yapılandırma dosyasıdır. Ön işlemci eklentisi olarak tasarlanan sistemin Snort tarafından çalıştırılabilmesi için bu dosyaya eklenmesi gerekmektedir. Ön işlemci eklentileri *snort.conf* dosyasındaki tanımlanma sıralarına göre çalıştırılırlar. Bu yüzden ön işlemci eklentisinin nereye ekleneceği Snort'un ve

tasarlanan eklentinin çalışma şeklini doğrudan etkilemektedir. Tasarlanan ön işlemci Snort'ta bulunan akış kontrolü, IP bütünleştirme, durum bilgili denetleme ve oturum yeniden kurma ön işlemcilerinden sonra çalışacak şekilde yapılandırılmıştır. Bu sıralamanın nedeni kendisinden daha önce çalışan ön işlemcilerin tasarlanan ön işlemci için ön gereksinimleri tamamlıyor olmasıdır. *Snort.conf* dosyasından sisteme çalışma kipleri ve algoritma parametreleri iletilmektedir. *Snort.conf* dosyasına Şekil 3.2'deki ifadeler eklenmiştir.

```
# --get-normal
# --get-bad
# --svm-train
# --svm-test
# --scale : Normalize data, 0 or 1 (default 0)
# --session-timeout : Session Timeout in seconds (default 60)
# --max_conn : Maximum Concurrent Connections (default 1000)
# --window-size : Session Window value (default 100)
# --window-time : Time Window value in seconds (default 2)
#
# -svm_type : set type of SVM (default 0)
# 0 -- C-SVC
# 1 -- nu-SVC
# 2 -- one-class SVM
# 3 -- epsilon-SVR
# 4 -- nu-SVR
# -kernel_type : set type of kernel function (default 2)
# 0 -- linear: u'*v
# 1 -- polynomial: (gamma*u'*v + coef0)^degree
# 2 -- radial basis function: exp(-gamma*|u-v|^2)
# 3 -- sigmoid: tanh(gamma*u'*v + coef0)
# 4 -- precomputed kernel (kernel values in training_set_file)
# -degree : set degree in kernel function (default 3)
# -gamma : set gamma in kernel function (default 1/k)
# -cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
# -epsilon : set tolerance of termination criterion (default 0.001)
# -probability_estimates: whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
#
preprocessor svm: --get-bad
```

Şekil 3.2 Snort.conf yapılandırma dosyası.

- **Debug.h:** Sistem hata ayıklama kipinde çalıştırılacaksa bu dosyada tanımlanan değer kullanılarak sistem için hata ayıklama aktif hale getirilebilir. Sistem için girilen değer aşağıdaki gibidir.

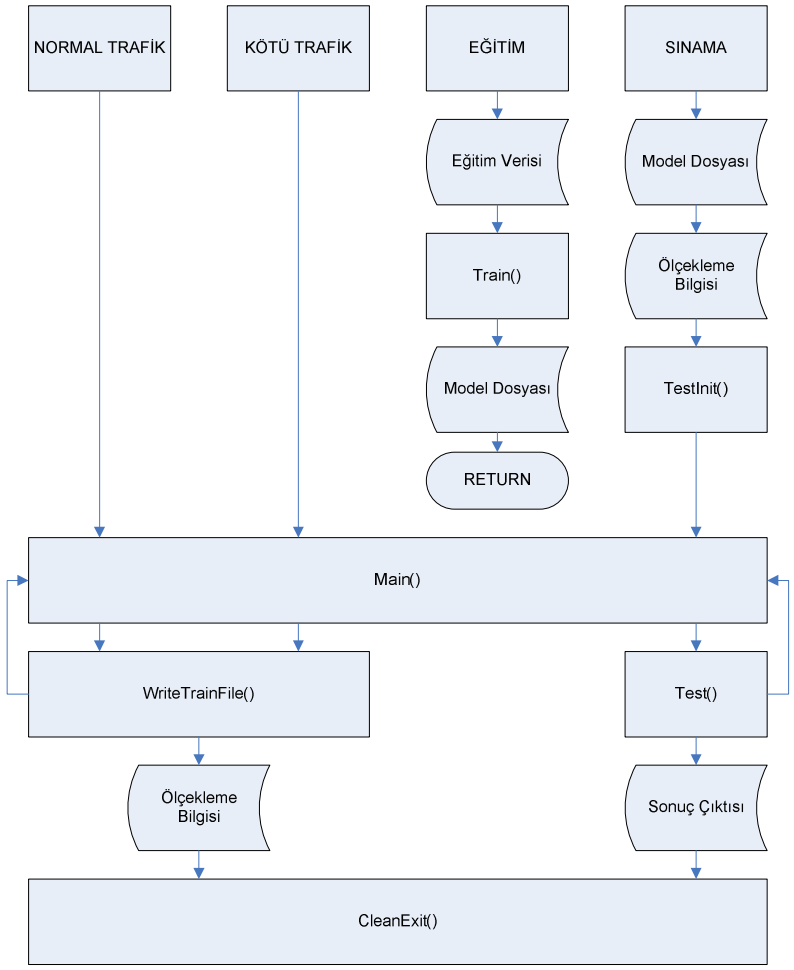
```
#define DEBUG_SVM             0x20000000  /* 536870912 */
```

Eğer sistem hata ayıklama kipinde derlenmişse, Snort çalıştırılmadan önce `SNORT_DEBUG` değeri sistem değişkeni olarak eklenmelidir. Aşağıdaki komut çalıştırıldığında sistem hata ayıklama kipinde çalışacaktır.

```
export SNORT_DEBUG = 536870912
```

Sistemin 4 farklı çalışma kipi vardır. Bu kipler snort.conf dosyası üzerinde değiştirilmektedir. Şekil 3.3'de kipler ve genel akışları gösterilmiştir.

1. Normal Trafiğin Analizi
2. Kötü Trafiğin Analizi
3. Eğitim
4. Sınama



Şekil 3.3 Sistemin çalışma kipleri ve genel akışları.

3.1.1 Normal Trafiğin Analizi

Bu kipte, gelen trafik "normal" olarak kabul edilir. Paketler analiz edilerek Bölüm 3.3'de belirtilen öznitelikler çıkartılır. Eğitim için dosyaya yazılan değerler, istenirse [0,1] aralığına ölçeklenir.

3.1.2 Kötü Trafiğin Analizi

Bu kipte, gelen trafik "kötü" olarak kabul edilir. Paketler analiz edilerek Bölüm 3.3'de belirtilen öznitelikler çıkartılır. Eğitim için dosyaya yazılan değerler, istenirse [0,1] aralığına ölçeklenir.

3.1.3 Eğitim

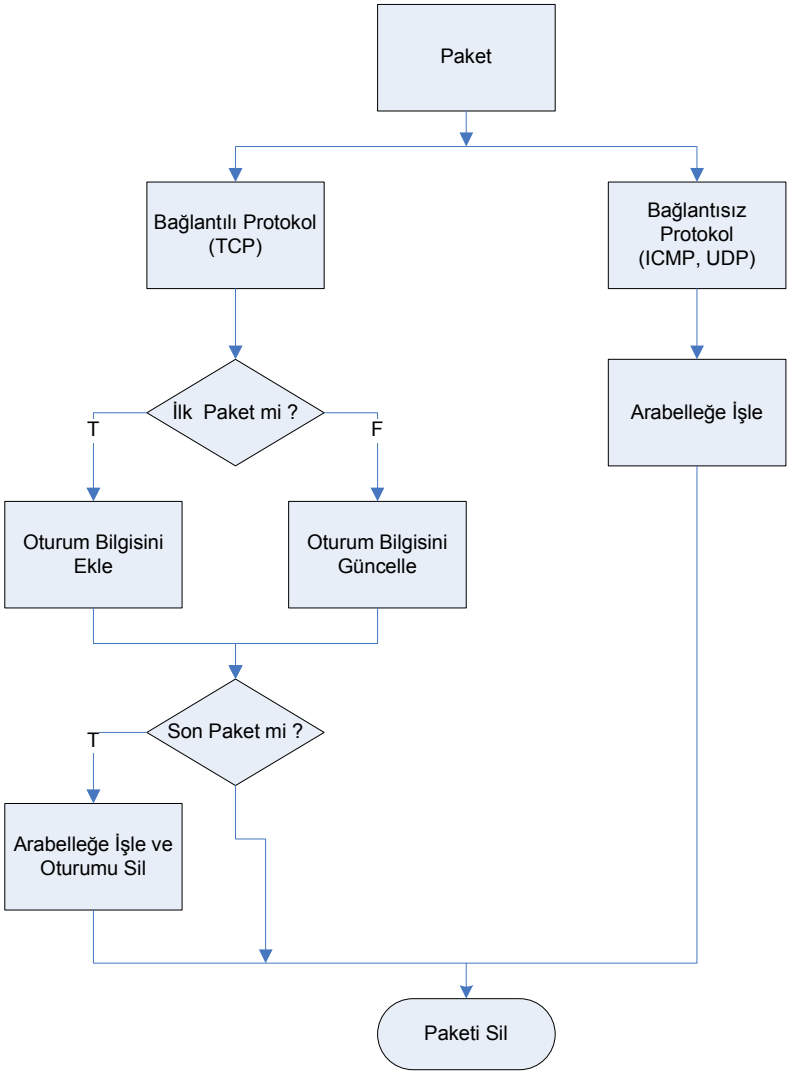
Normal ve kötü trafiğin analizinden elde edilen öznitelikler, eğitim kipinde, girdi olarak sisteme verilir. Eğitim algoritması çalıştırılarak üretilen değerler model dosyasına yazılır.

3.1.4 Sınama

Eğitim kipinden gelen model dosyası ve trafik analizinden gelen ölçekleme bilgisi sisteme girdi olarak verilir. Sistem, bilgisayar ağı trafiğinden gelen paketleri gerçek zamanlı olarak analiz ederek Bölüm 3.3'de belirtilen öznitelikleri çıkartır. Analiz sonuçları sınıflandırma algoritmasına verilerek sınanır. Test sonucunda bir anormallik tespit edilmişse Snort'un çıkış eklentisine durum iletilerek alarm verilir. Ayrıca, tüm sonuçlar doğruluk oranını değerlendirmek amacıyla bir dosyaya kaydedilir.

3.2 Sistemin Veri Akış Yapısı

Tasarlanan sistemin temel girdisi bilgisayar ağı paketleridir. Her ağ paketi için sistemdeki ana fonksiyon çağırılmaktadır. Paketin tipine ve seçilen çalışma kipine göre paket Şekil 3.4'te gösterilen akışı takip etmektedir.



Şekil 3.4 Ağ paketinin sistemdeki akışına ait diyagram.

3.3 Tanıma İçin Kullanılan Öznitelikler

Tasarlanan sistem, bilgisayar ağı trafiğinden tanımayı sağlayacak belirgin öznitelikler çıkarmaktadır. Lee W.'nin makalesinde, bu öznitelikler elde edilmeleri için gereken işlem zamanlarına göre seviyelendirilmişlerdir (Lee et al., 2001). 1. seviye öznitelikler tek bir ağ paketi bilgisinden elde edilebilirler (Örn: IP numarası). 2. seviye öznitelikler bir oturum süresince herhangi bir zamanda değişebilirler (Örn: Gönderilen veri miktarı). 3. seviye öznitelikler oturum sonunda hesaplanabilirler (Örn: Oturum süresi). 4. seviye öznitelikler de oturum sonunda hesaplanırlar, ancak hesaplanmaları için daha önceki oturum bilgilerinin de

bilinmesi gerekir (Örn: *t* süresi boyunca aynı kaynaktan gelen bağlantı sayısı). 4. seviye öznitelikler daha genel bilgi verirler ancak işlem zamanları yüksektir.

Sistem tasarımında farklı seviyelerdeki özniteliklerin kullanımı sistem başarımını etkilemektedir. Hem çalışma hızı hem de doğruluk göz önüne alınarak kabul edilebilir sınırlarda uygun öznitelikler seçilerek uygun değerlerde bir sistem tasarlanmak amaçlanmıştır.

Sistem'de kullanılan öznitelikler 3. Uluslararası Bilgi Keşfi ve Veri Madenciliği Araçları Yarışması'ndan (KDD Cup'99) [13] alınmıştır. Yarışmanın amacı anormallik tespiti ile tanıma yapan ağ saldırı tespit sistemlerinin başarı oranlarını ölçmektir. Bu yarışmada kullanılan veri kümesi DARPA (Defense Advanced Research Projects Agency) tarafından 1999 yılında yapılan bilgisayar ağı saldırı benzetiminde oluşturulmuş verilerdir. DARPA veri kümesi saldırı tespit sistemlerinin ölçümü için kullanılan standart bir veri kümesidir. Tasarlanan sistemin eğitimi ve sınanması için bu veri kümesinden yararlanılmıştır. Çizelge 3.1'de kullanılan öznitelikler ve açıklamaları verilmiştir.

Gri olarak işaretlenmiş öznitelikleri elde etmek için sunucu seviyesinde bilgi gerekmektedir. Bu nedenle bilgisayar ağından elde edilen bilgiler ile bu öznitelikler oluşturulamamaktadır. Ayrık tipteki öznitelikler tek bir ağ paketi bilgisinden elde edilebilirken sürekli tiptekiler için bir veya birden fazla oturum bilgisine ihtiyaç vardır. Sürekli tipteki öznitelikler, tek bir oturum bilgisi ile belirlenebileceği gibi daha önceki oturumlara ait bilgilere de ihtiyaç duyabilirler. Daha önceki oturum bilgilerini tutmak için iki farklı bellek yapısı kullanılmaktadır. İlk bellek yapısında, *t* saniye içerisinde gerçekleşen oturumlar tutulmaktadır. İkinci yapıda ise en son gerçekleşen *n* adet oturum bilgisi saklanmaktadır. Bu bellekler, sistemin bilgisayar ağı trafiği hakkında daha genel bilgi elde etmesini sağlarlar. Sınama aşamasında da değinileceği üzere *t* ve *n* değerlerinin seçimi sistem başarımını doğrudan etkilemektedir.

Çizelge 3.1 Anormallik tespitinde kullanılan öznitelikler.

Özellik	***Açıklama***	***Tip***
duration	bağlantı süresi (sn.)	sürekli
protocol_type	protokol tipi, örn. tcp, udp, vb.	ayrık
service	hedef ağ servisi, örn., http, telnet, vb.	ayrık
src_bytes	kaynaktan hedefe iletilen veri miktarı (byte)	sürekli
dst_bytes	hedeften kaynağa iletilen veri miktarı (byte)	sürekli
flag	bağlantının durumu: normal veya hata	ayrık
land	bağlantının kaynak ve hedefi aynı sunucu/port ise 1; diğer 0	ayrık
wrong_fragment	hatalı parça sayısı	sürekli
urgent	acil paket sayısı	sürekli
hot	"sıcak" göstergelerin sayısı	sürekli
num_failed_logins	hatalı giriş sayısı	sürekli
logged_in	başarılı giriş 1; diğer 0	sürekli
num_compromised	"tehlikeli" durum sayısı	sürekli
root_shell	root shell ele geçirldiyse 1; diğer 0	ayrık
su_attempted	``su root'' komutu çalıştırılıdıysa 1; diğer 0	ayrık
num_root	"root" erişim sayısı	sürekli
num_file_creations	yaratılan dosya sayısı	sürekli
num_shells	açık shell sayısı	sürekli
num_access_files	erişim kontrol dosyalarındaki işlem sayısı	sürekli
num_outbound_cmds	ftp oturumundaki giden komut sayısı	sürekli
is_hot_login	"sıcak" listeden bir giriş ise 1; diğer 0	ayrık
is_guest_login	"misafir" girşi ise 1; diğer 0	ayrık
Aşağıdaki öznitelikler aynı sunucuya t sn süresince yapılan bağlantıları işaret eder.		
count	aynı sunucuya t saniye içinde yapılan bağlantı sayısı	sürekli
serror_rate	``SYN'' hatası alınan bağlantı yüzdesi	sürekli
rerror_rate	``REJ'' hatası alınan bağlantı yüzdesi	sürekli
same_srv_rate	aynı servise yapılan bağlantı yüzdesi	sürekli
diff_srv_rate	farklı servislere yapılan bağlantı yüzdesi	sürekli
Aşağıdaki öznitelikler aynı sunucuya n adet bağlantı içinde yapılan bağlantıları işaret eder.		
dst_host_count	aynı sunucuya n adet bağlantı içinde yapılan bağlantı sayısı	sürekli
dst_host_serror_rate	``SYN'' hatası alınan bağlantı yüzdesi	sürekli
dst_host_rerror_rate	``REJ'' hatası alınan bağlantı yüzdesi	sürekli
dst_host_same_srv_rate	aynı servise yapılan bağlantı yüzdesi	sürekli
dst_host_diff_srv_rate	farklı servislere yapılan bağlantı yüzdesi	sürekli
dst_host_same_src_port_rate	aynı kaynak portundan gelen bağlantı yüzdesi	sürekli
Aşağıdaki öznitelikler aynı servise t sn süresince yapılan bağlantıları işaret eder.		
srv_count	aynı servise t saniye içinde yapılan bağlantı sayısı	sürekli
srv_serror_rate	``SYN'' hatası alınan bağlantı yüzdesi	sürekli
srv_rerror_rate	``REJ'' hatası alınan bağlantı yüzdesi	sürekli
srv_diff_host_rate	farklı sunuculara yapılan bağlantı yüzdesi	sürekli
Aşağıdaki öznitelikler aynı servise n adet bağlantı içinde yapılan bağlantıları işaret eder.		
dst_host_srv_count	aynı servise n adet bağlantı içinde yapılan bağlantı sayısı	sürekli
dst_host_srv_serror_rate	``SYN'' hatası alınan bağlantı yüzdesi	sürekli
dst_host_srv_rerror_rate	``REJ'' hatası alınan bağlantı yüzdesi	sürekli
dst_host_srv_diff_host_rate	farklı sunuculara yapılan bağlantı yüzdesi	sürekli

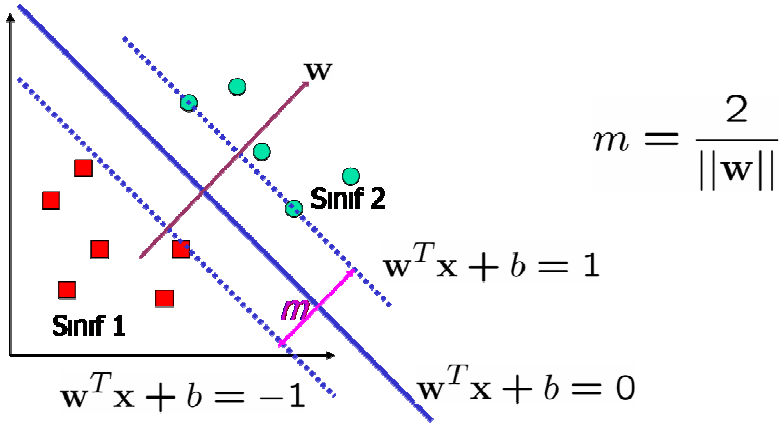
3.4 Anormallik Tespiti Ve Destek Vektör Makinesi (SVM)

Sistemin ikinci parçası anormallik tespiti algoritmasından oluşmaktadır. Bu parça eğitim ve sınama olarak iki alt bölümde incelenebilir. Eğitim kısmında, normal ve kötü trafik bilgileri sisteme verilerek öğrenme gerçekleştirilir. Sınama kısmında, bilgisayar ağ trafiğinden

öznitelikler çıkartılmakta ve eğitim aşamasında elde edilen sonuçlar ile gerçek zamanlı olarak kıyaslanmaktadır. Kıyaslama sonucunda oluşan alarmlar çıkış eklentilerine bildirilmektedir. Programın hangi durumda çalışacağı bir yapılandırma dosyası üzerinden kontrol edilmektedir. Ayrıca öğrenme algoritmasına ait parametreler de yapılandırma dosyası aracılığıyla programa verilmektedir.

Sistemde, sınıflandırma algoritması olarak destek vektör makinesi (SVM) kullanılmıştır. Ancak, farklı sınıflandırma algoritmaları ile de (Random Forest, J48, Naive Bayes, NBTree, KNN, MLP, SOM) sistem sınanmıştır. Mukkamala S.'in makalesinde de belirttiği üzere destek vektör makinesi saldırı tespitinde yapay sinir ağlarına göre hem daha doğru hem de daha hızlı sonuçlar üretmektedir (Mukkamala et al., 2002). SVM ikili sınıflandırma yapmaktadır. Sistemde normal ve kötü trafik olmak üzere iki adet sınıf olduğu için SVM algoritması için uygundur.

SVM, değişik çekirdek fonksiyonları ile sınıflandırma ve bağlanım için istatistiksel öğrenme tekniklerini kullanan bir çeşit örüntü sınıflandırıcısıdır. Şekil 3.5'de görüldüğü üzere, Farz edelim ki pozitif ve negatif örnekleri birbirinden ayıran bir aşırı düzlem var, bu düzlem üzerindeki noktalar w.x+b=0 eşitliğini sağlayacaktır, burada w aşırı düzleme olan normal ve |b|/||w|| aşırı düzlemden orijine olan dik uzaklıktır. Aşırı düzleme en yakın pozitif ve negatif örnekler arasındaki mesafeye ayırıcı aşırı düzleminin "tolerans"ı dersek, destek vektör yöntemi bu "tolerans"ın en yüksek olduğu bir aşırı düzlem bulmaya çalışır (Kepenekci & Akar, 2004).



Şekil 3.5 Destek Vektör Makinesi.

Çekirdek fonksiyonları, problemi bir üst uzaya taşıyarak tanımayı kolaylaştırırlar. Çoğunlukla kullanılan SVM çekirdek fonksiyonları aşağıdaki gibidir:

- Doğrusal : $K(x_i, x_j) = x_i^T x_j$
- Polinomsal : $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
- Radyal taban fonksiyonu : $K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2), \gamma > 0$
- Sigmoid fonksiyonu : $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Problemin türüne göre kullanılacak çekirdek fonksiyonu değişkenlik gösterir. Probleme uygun seçilecek çekirdek fonksiyonu sistemin doğruluk oranını önemli ölçüde etkiler.

SVM algoritmasının uygulanmasında, C++ programlama dilinde yazılmış olan LibSVM [15] kütüphanesinden yararlanılmıştır. Hızlı çalışması ve farklı SVM algoritmalarını içermesi nedeniyle uygulamada LibSVM kütüphanesi tercih edilmiştir. Ayrıca, LibSVM'de parametre seçimi için yararlı araçlar bulunmaktadır. Daha önce denenen SvmLight [16] aracı LibSVM gibi bir kütüphane olarak hazırlanmadığı için tasarlanan sistem ile birlikte derlemek mümkün olmamaktadır. Ayrı bir program olarak sistem tarafından çağrılması gerekmektedir. Bu durum hem sistemin çalışma hızını olumsuz etkilediğinden hem de Snort'un standart yapısını bozacağından SvmLight aracının kullanılmasından vazgeçilmiştir.

3.5 Sistemin Geliştirilme Ortamı

Sistem, Linux işletim sistemi üzerinde geliştirilmiştir. Güvenirliliği, kararlı çalışması, başarımı, ölçeklenebilirliği, maliyeti, açık kaynak kod özelliği ve bilişim dünyasında yaygınlığı göz önünde bulundurularak uygulama platformu olarak Linux işletim sistemi seçilmiştir. Sistemin geliştirildiği platformun özellikleri Çizelge 3.2'de belirtilmiştir.

Çizelge 3.2 Sistemin geliştirilme ortamı

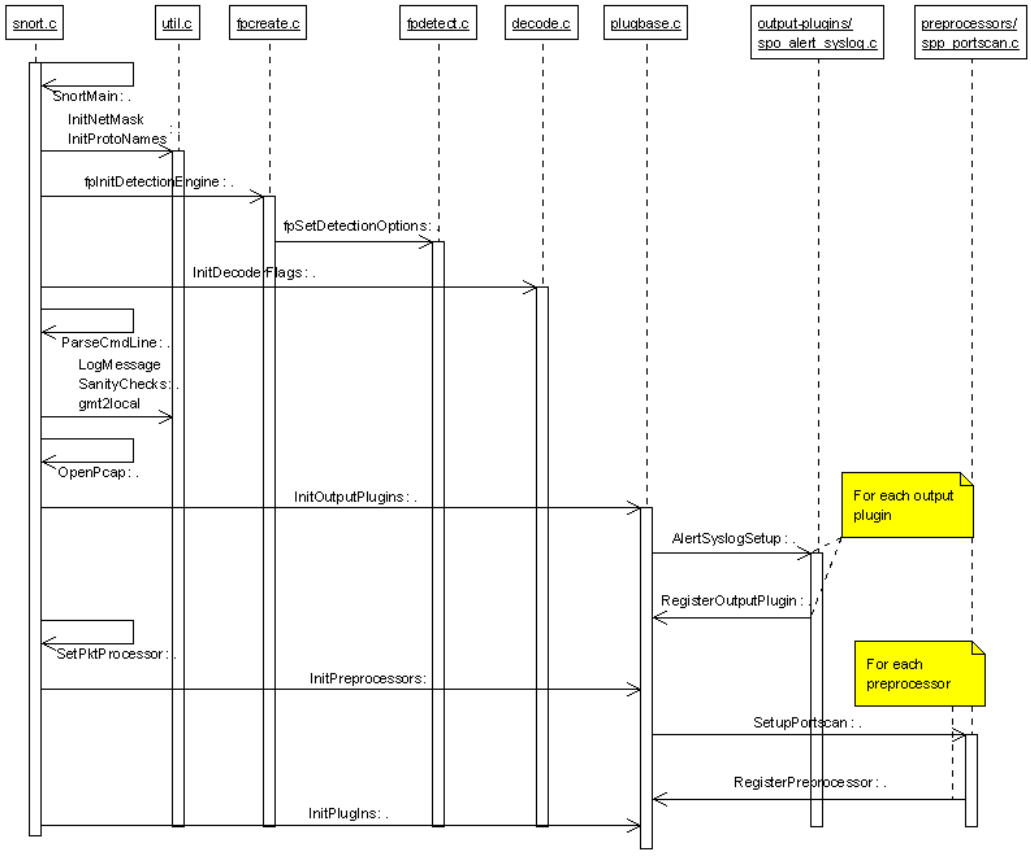
Sistem	Dell Latitude D600
İşlemci	Intel(R) Pentium(R) M processor 1.60GHz
Birincil Bellek	768 MB
İkincil Bellek	30 GB
İşletim Sistemi	OpenSuse 10.2 [17]
Derleyici	GNU GCC 4.1 [18]
Diğer Yazılımlar	
	Snort 2.6.1.2 [3]
	LibSvm 2.83 [15]

4. KULLANILAN FONKSİYONLAR

Sistemde gerçeklenen fonksiyonlar işlevlerine göre aşağıda belirtilmişlerdir.

4.1 Snort'a Ait Fonksiyonlar

Tasarlanan sistem, Snort saldırı tespiti yazılımına önişlemci eklentisi olarak yazıldığı için bazı Snort fonksiyonları ile etkileşim halindedir. Bu bölümde tasarlanan sistemin Snort ile birlikte çalışması sırasında çağırılan Snort fonksiyonları incelenmiştir. Şekil 4.1'de Snort programının açılış aşaması gösterilmektedir. Snort, *Initpreprocessors* fonksiyonunu çalıştırarak *plugbase.c* dosyasındaki önişlemci açılış fonksiyonlarını sırayla çağırmakta ve önişlemcileri sisteme eklemektedir.



Şekil 4.1 Snort programının açılış diyagramı (Arboleda & Bedón, 2006).

4.1.1 InitPreprocessors Fonksiyonu

Snort'un açılışında çağrılır. Görevi, önişlemcileri sırayla etkinleştirmek ve çalışmaya hazır hale getirmektir. Fonksiyon, tasarlanan sisteme ait SetupSvm() fonksiyonunu çağırır.

4.1.2 RegisterPreprocessor Fonksiyonu

Her ön işlemci tarafından ilk çalıştırılma anında çağrılır. Ön işlemcilerin Snort'a kaydolmasını sağlar. Ön işlemci için gerekli başlangıç işlemleri yapılır.

4.1.3 AddFuncToPreprocList Fonksiyonu

Ön işlemcileri çalışma listesine ekler. Listede önişlemcilerin çalışma önceliği ve çağırılacak fonksiyonun adı bulunmaktadır. Snort, çalışma anında her paket için bu listedeki öncelik sırasına göre belirtilen fonksiyonları sırayla çağırmaktadır.

4.1.4 AddFuncToPreprocRestartList Fonksiyonu

Ön işlemcileri yeniden başlatma listesine ekler. Snort yeniden başlatılacaksa bu listedeki öncelik sırasına göre belirtilen fonksiyonları çağırılır. Dinamik hafıza alanları, açık dosyalar burada verilen fonksiyon içerisinde kapatılmalıdır.

4.1.5 AddFuncToPreprocCleanExitList Fonksiyonu

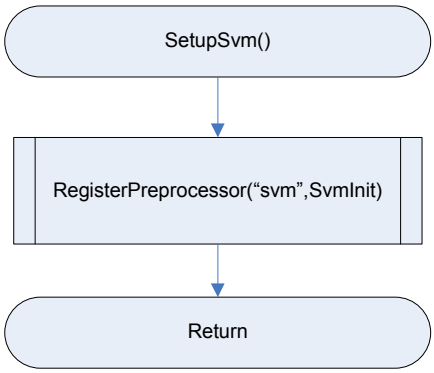
Ön işlemcileri çıkış listesine ekler. Snort kapatılacaksa bu listedeki öncelik sırasına göre belirtilen fonksiyonları çağırılır. Dinamik hafıza alanları, açık dosyalar burada verilen fonksiyon içerisinde kapatılmalıdır.

4.2 Başlangıç Fonksiyonları

Tasarlanan sistemin başlatılması esnasında kullanılan fonksiyonlardır.

4.2.1 SetupSvm Fonksiyonu

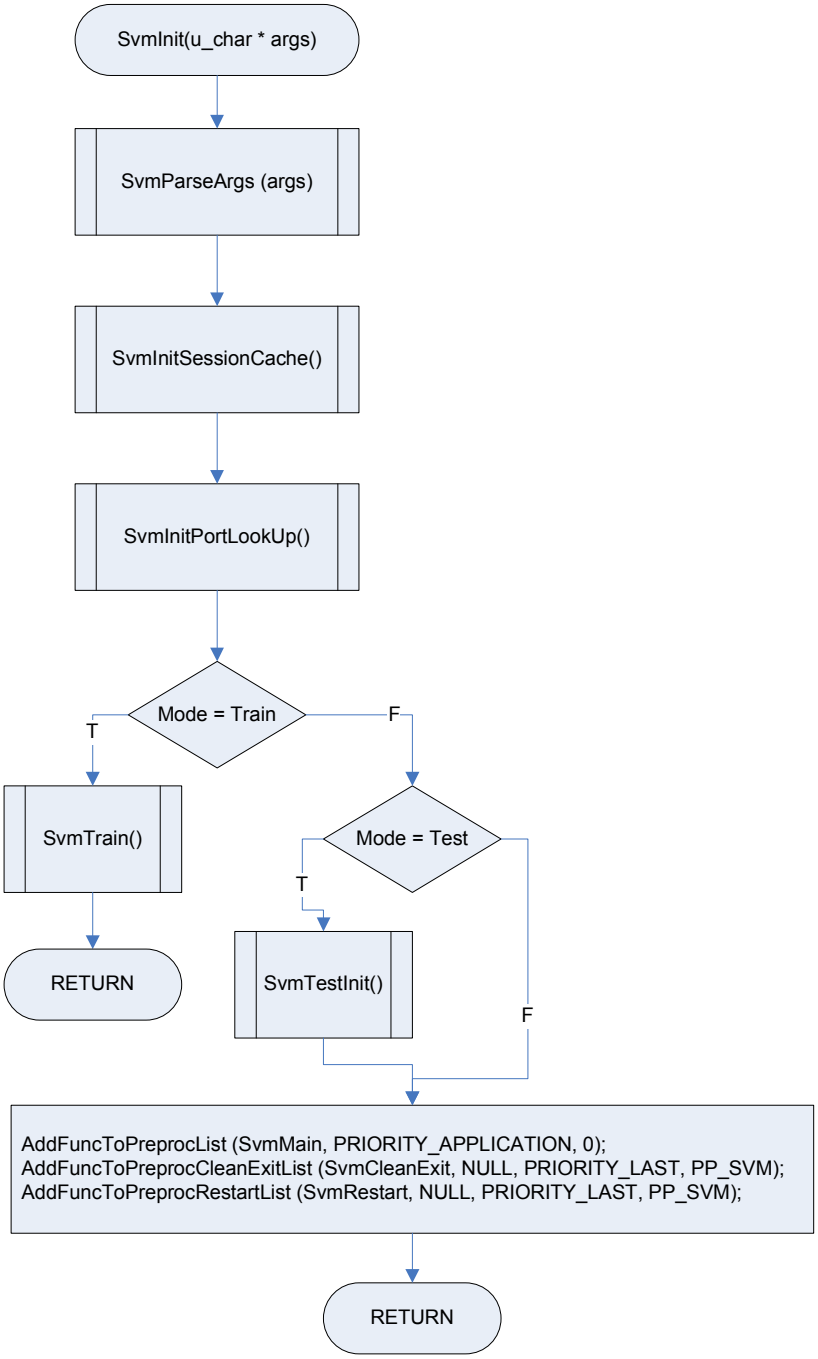
Snort'a ait InitPreprocessors() fonksiyonu tarafından çağrılır. Snort'a ait RegisterPreprocessor() fonksiyonunu çağırarak ön işlemcinin etkinleştirilmesini sağlar. Snort'un açılışında bir kez çağrılır. Şekil 4.2'de fonksiyonun akış diyagramı verilmiştir.



Şekil 4.2 SetupSvm fonksiyonu

4.2.2 SvmInit Fonksiyonu

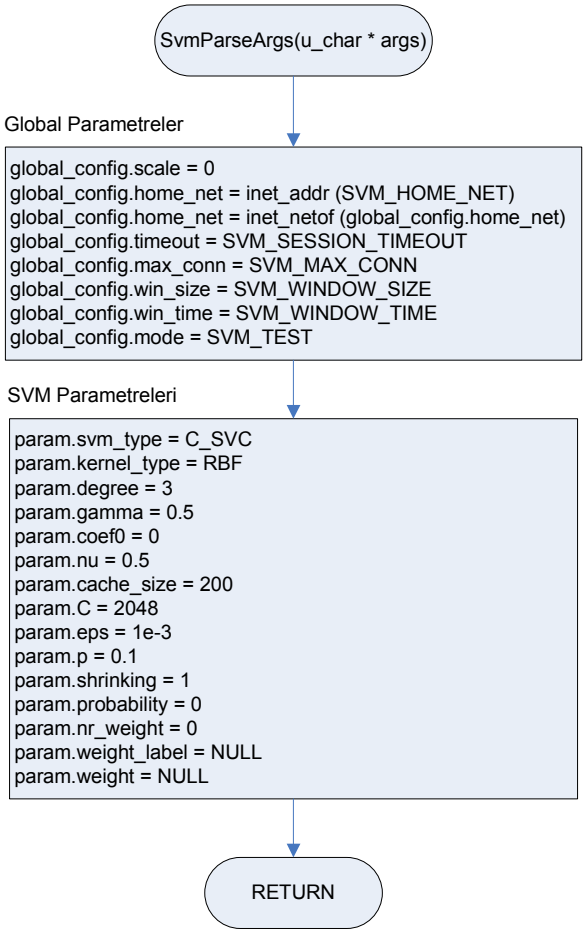
Sistemin başlatılması esnasında çalıştırılır. İşlevi, programın kullanacağı değişken veya yapıları oluşturmak, yapılandırma dosyasını okuyarak programın işleyişine karar vermektir. Şekil 4.3'de fonksiyonun akış diyagramı verilmiştir.



Şekil 4.3 SvmInit fonksiyonu

4.2.3 SvmParseArgs Fonksiyonu

SvmInit fonksiyonu tarafından çağrılır. Snort.conf yapılandırma dosyasından gelen ön işlemci parametrelerini çözümler. Şekil 4.4'te sadece varsayılan değerler gösterilmiştir.



Şekil 4.4 SvmParseArgs fonksiyonu ve varsayılan değerleri.

4.2.4 SvmInitSessionCache Fonksiyonu

Sistemde oturumlara ait durum bilgileri bir kıyım tablosunda (hash table) tutulmaktadır. SvmInitSessionCache fonksiyonu kıyım tablosunun boyutunu hesaplamakta ve hafızada uygun bir yer açmaktadır. Kıyım tablosunun boyutunu hesaplamak için, aynı anda kabul edilecek bağlantı sayısı 1.4 sabiti ile çarpılır. Bu sabit, kıyımlı arama sonucu çıkacak anlamlı rastlamaların üst düzeyde olması için kıyım tablosunun boyutunu genişletir. Kıyım algoritmasından çıkan değerlerin dağılımının eşit olabilmesi için tablo boyutunun bir asal sayı olması gerekir. Tablo boyutu hesaplanan değere eşit veya daha büyük en yakın asal sayı alınır. Tablo boyutu sabit olduğu için sisteme yapılabilecek bellek taşırma saldırılarının önüne geçilmiş olur.

4.2.5 SvmInitPortLookUp Fonksiyonu

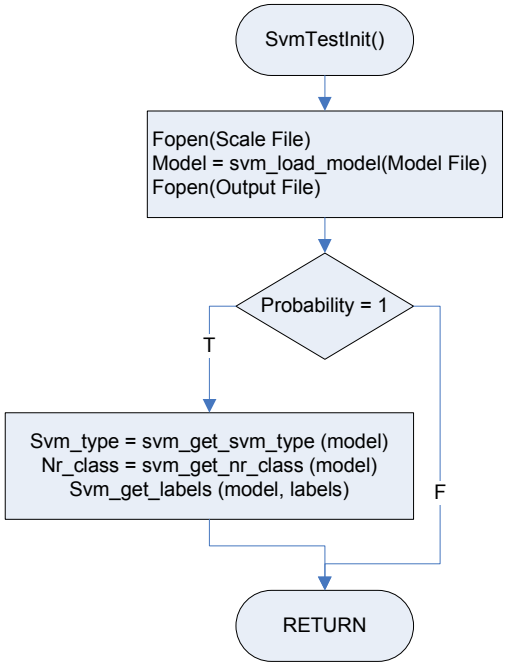
SvmInit() fonksiyonu tarafından çağrılır. Bilinen TCP/UDP port adresleri bir eşleştirme tablosunda saklanır. Bu tablonun kullanımı hem arama hem de doğruluk oranını arttırmaktadır. SvmInitPortLookUp fonksiyonu eşleştirme tablosunu oluşturmaktadır. Şekil 4.5'te eşleştirme tablosundan birkaç örnek verilmiştir.

```
port_lookup[80] = maxport++;      //http
port_lookup[20] = maxport++;      //ftp-data
port_lookup[21] = maxport++;      //ftp
port_lookup[22] = maxport++;      //ssh
port_lookup[23] = maxport++;      //telnet
port_lookup[25] = maxport++;      //smtp
port_lookup[53] = maxport++;      //domain
```

Şekil 4.5 SvmInitPortLookUp fonksiyonu

4.2.6 SvmTestInit Fonksiyonu

SvmInit fonksiyonu tarafından çağrılır. Eğer sistem sınama kipinde başlatıldı ise SvmTestInit fonksiyonu sınama için gerekli veri yapılarını ve değişkenleri hazırlar. Model ve ölçekleme dosyalarını okur. Çıkış dosyasını yazma yönünde hazır hale getirir. Sınama sonucunun olasılıklı olup olmayacağı değerlendirilir. Şekil 4.6'da fonksiyonun akış diyagramı verilmiştir.



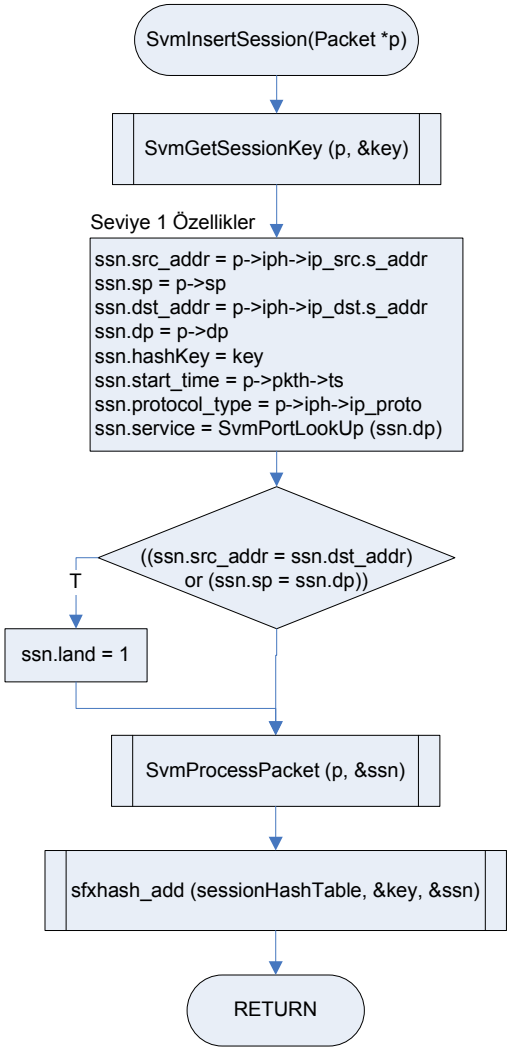
Şekil 4.6 SvmTestInit fonksiyonu.

4.3 Oturum Durum Bilgisini Düzenleyen Fonksiyonlar

Geliştirilen sistem TCP oturumları için durum bilgili analiz yapmaktadır. Durum bilgilerinin saklanması ve yönetilmesi için kıyım tablosundan (hash table) yararlanılmıştır. Bu durum her paket için ait olduğu oturumu aramayı hızlandırmakta ve doğrudan sistem başarımına etki etmektedir. Daha önce bölüm 4.2.4'te bahsedilen SvmInitSessionCache fonksiyonu da bu grupta incelenebilir.

4.3.1 SvmInsertSession Fonksiyonu

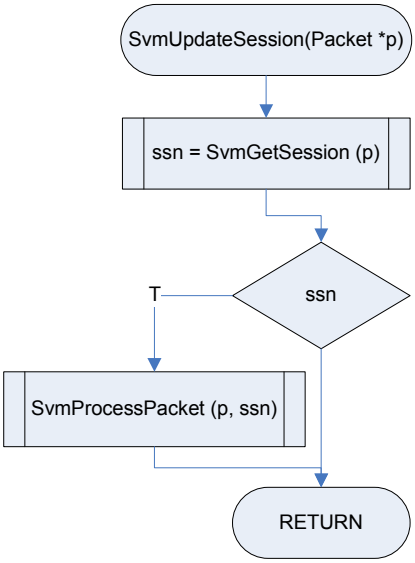
Yeni bir oturum bağlantısı tespit edildiğinde SvmInsertSession fonksiyonu çağrılarak yeni oturum kıyım tablosuna eklenir. Bölüm 3.3'de değinilmiş olan 1. seviye öznitelik değerleri bu fonksiyon içinde hesaplanır. Şekil 4.7'de fonksiyonun akış diyagramı verilmiştir.



Şekil 4.7 SvmInsertSession fonksiyonu

4.3.2 SvmUpdateSession Fonksiyonu

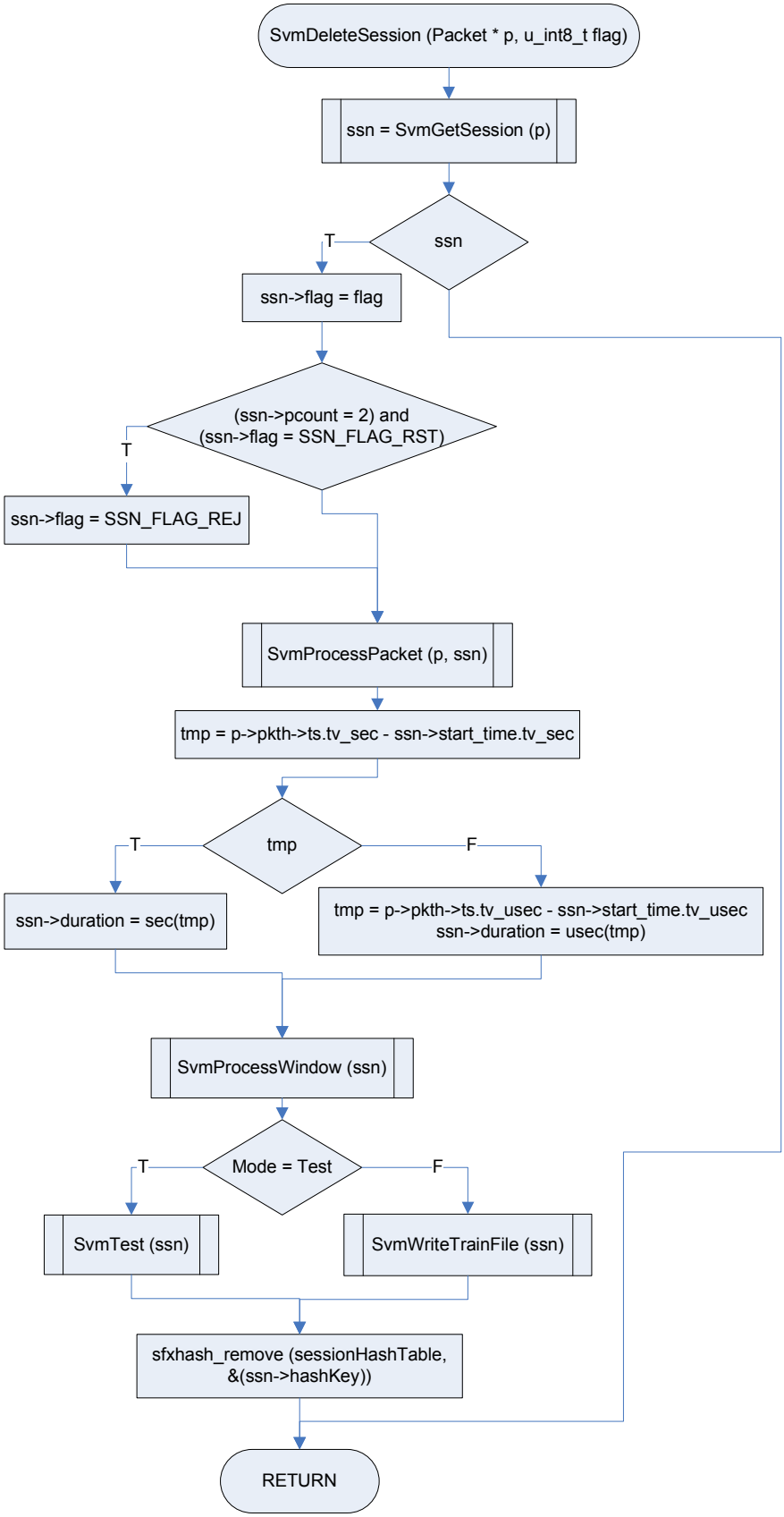
Oturum durum bilgisini günceller. Şekil 4.8'de fonksiyonun akış diyagramı verilmiştir.



Şekil 4.8 SvmUpdateSession fonksiyonu.

4.3.3 SvmDeleteSession Fonksiyonu

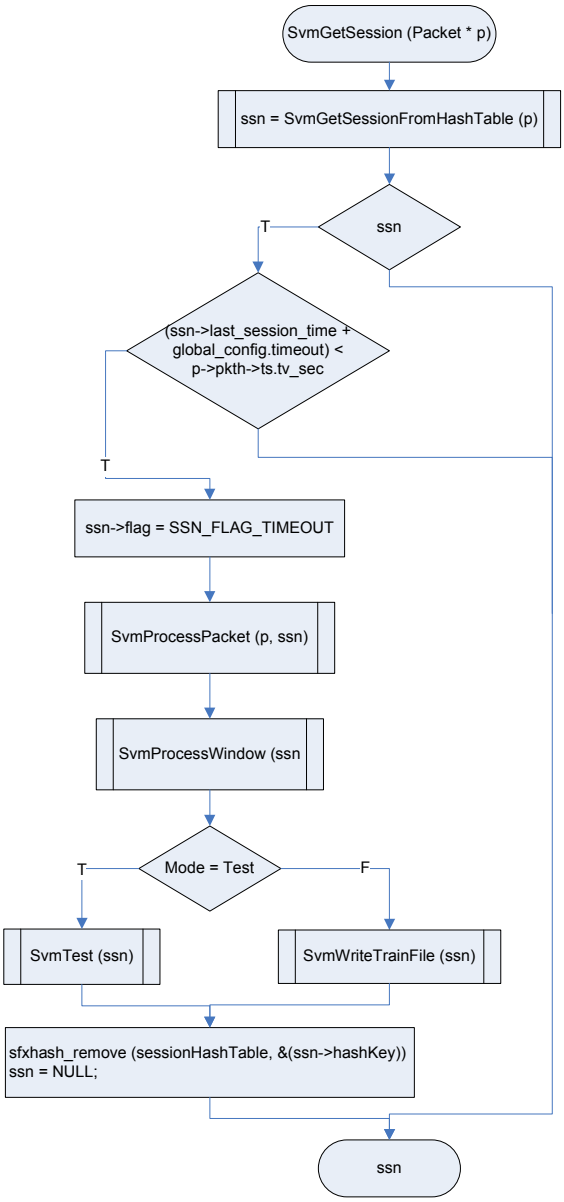
Bir oturum sonlandırıldığında, elde edilen durum bilgisinden öznitelikleri hesaplar ve daha sonra kıyım tablosundan oturum bilgisini siler. Şekil 4.9'da fonksiyonun akış diyagramı verilmiştir.



Şekil 4.9 SvmDeleteSession fonksiyonu

4.3.4 SvmGetSession Fonksiyonu

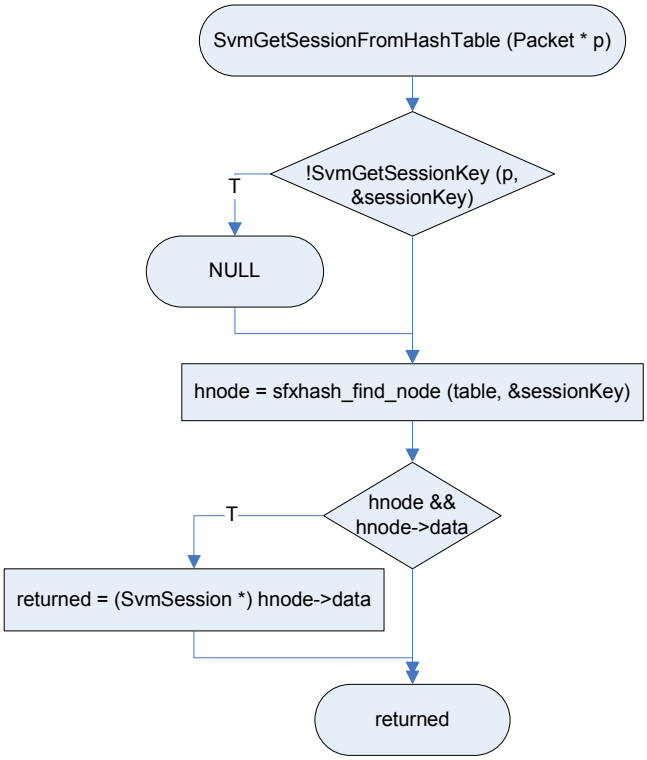
Paketin ait olduğu oturum bilgisini kıyım tablosundan getirir. Oturum zaman aşımına uğrayıp uğramadığını kontrol eder. Eğer zaman aşımı varsa oturumu kıyım tablosundan kaldırır. Şekil 4.10'da fonksiyonun akış diyagramı verilmiştir.



Şekil 4.10 SvmGetSession fonksiyonu.

4.3.5 SvmGetSessionFromHashTable Fonksiyonu

Paketin ait olduğu oturum bilgisini kıyım tablosunda arar. Eğer bulduysa oturuma ait veri yapısını döndürür. Şekil 4.11'de fonksiyonun akış diyagramı verilmiştir.



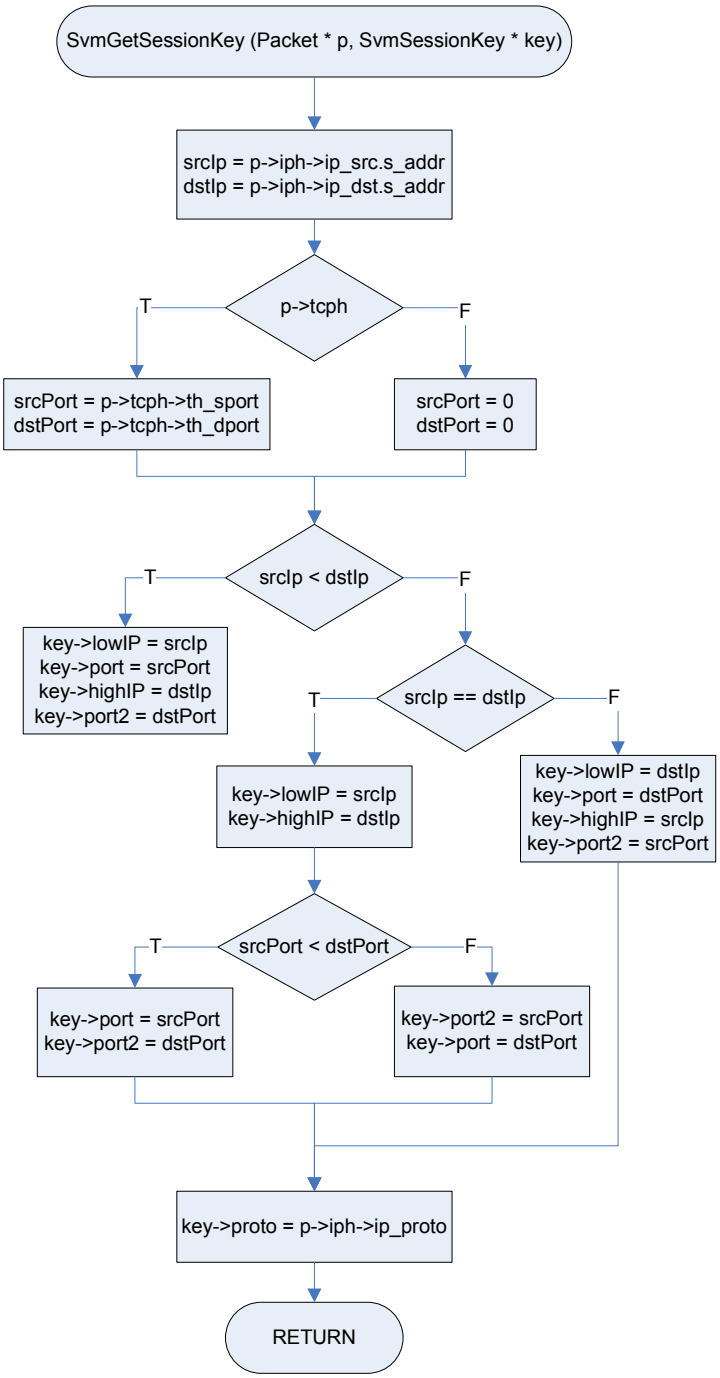
Şekil 4.11 SvmGetSessionFromHashTable fonksiyonu.

4.3.6 SvmGetSessionKey Fonksiyonu

Kıyım değerini hesaplar. Kıyım değeri aşağıda belirtilen değişkenlerden oluşan bir veri yapısıdır.

- Kaynak IP adresi
- Kaynak port numarası
- Hedef IP adresi
- Hedef port numarası
- Protokol numarası

Bu değişkenlerden yararlanarak her oturum için eşsiz bir kıyım değeri hesaplanmaktadır. Şekil 4.12'de fonksiyonun akış diyagramı verilmiştir.



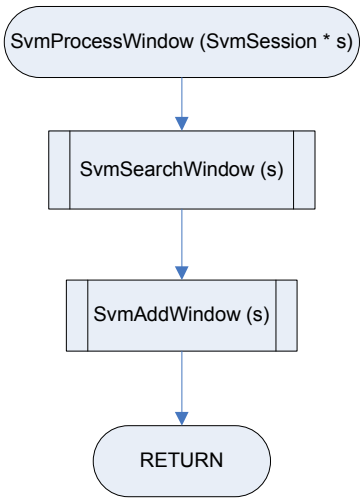
Şekil 4.12 SvmGetSessionKey fonksiyonu.

4.4 Arabellek Yapısını Düzenleyen Fonksiyonlar

Bölüm 3.3'de belirtilen 4. seviye öznitelikleri hesaplamak için daha önceki ağ bağlantılarına ait bilgilere ihtiyaç vardır. Geçmiş ağ bağlantılarına ait bilgiler sistemde bir arabellek yapısında tutulmaktadır. Arabellek iki farklı yaklaşım ile kullanılmaktadır. İlk yaklaşımda, arabellekte, t sn. içerisinde yapılmış olan ağ bağlantıları tutulmakta; ikinci yaklaşımda ise son n adet ağ bağlantısı saklanmaktadır. Bu iki yaklaşım farklı saldırı tiplerinin tespitinde kullanılmaktadır. Zamana bağlı yaklaşımda, port taramalar, syn sağanağı gibi az bir zaman aralığında fazla miktarda paket gönderilen saldırı çeşitleri tespit edilebilmektedir. Niceliğe bağlı yaklaşımda ise uzun zamana yayılan saldırılar yakalanabilmektedir.

4.4.1 SvmProcessWindow Fonksiyonu

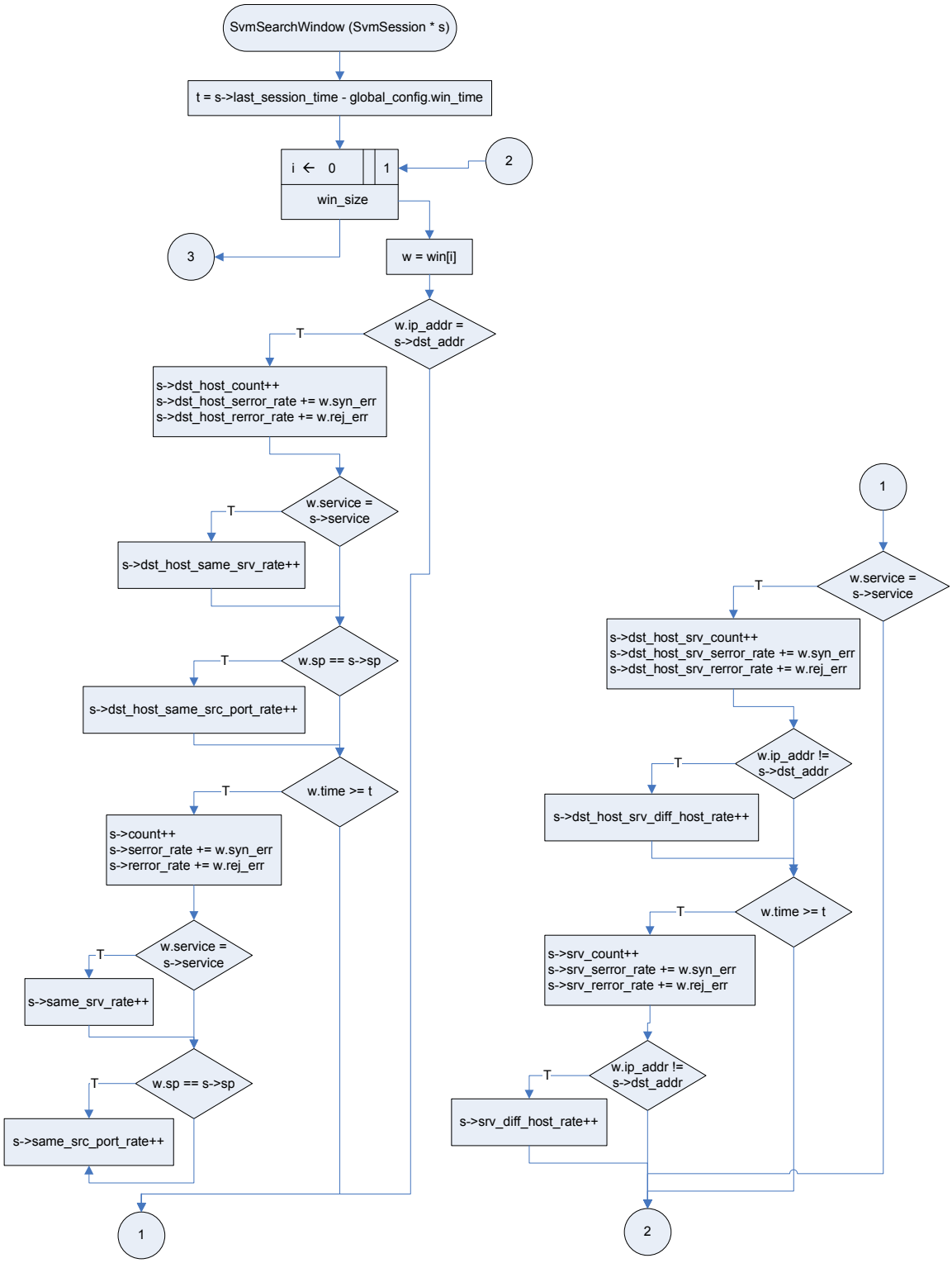
Gelen oturum bilgisini arabelleğe işler. Şekil 4.13'te fonksiyonun akış diyagramı verilmiştir.



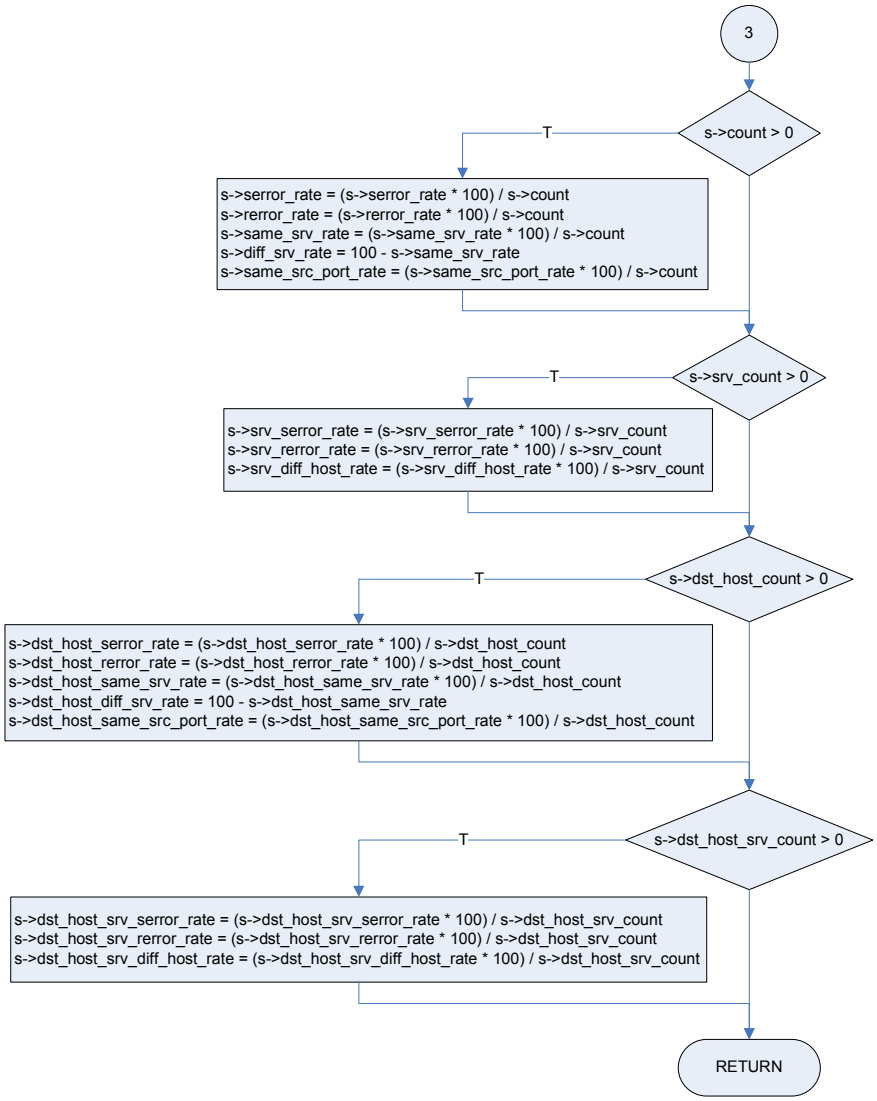
Şekil 4.13 SvmProcessWindow fonksiyonu.

4.4.2 SvmSearchWindow Fonksiyonu

Arabellekte arama yaparak öznitelik değerlerini hesaplar. Şekil 4.14 ve Şekil 4.15'de fonksiyonun akış diyagramları verilmiştir.



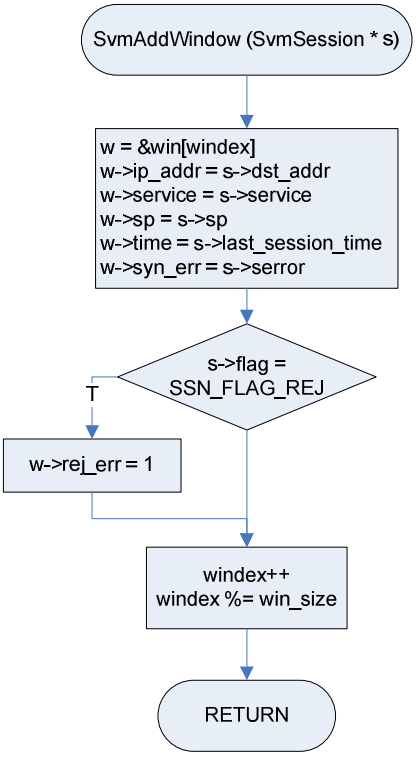
Şekil 4.14 SvmSearchWindow fonksiyonu (1).



Şekil 4.15 SvmSearchWindow fonksiyonu (2).

4.4.3 SvmAddWindow Fonksiyonu

Arabelleğe bir kayıt ekler. Arabelleğin yapısı çevrimsel sıralı olarak tasarlanmıştır. Böylece sisteme yapılacak hafıza taşırma saldırılarına karşı önlem alınmıştır. Dikkat edilirse, arabellek fonksiyonlarının içinde silme fonksiyonu yoktur. Arabellek yapısının çevrimsel sıralı oluşu silme işlemini gereksiz kılmıştır. Yeni gelen bilgi en eski bilginin üstüne yazılmaktadır. Burada önemli olan arabellek boyutunun yeterince büyük seçilmesidir. Şekil 4.16'da fonksiyonun akış diyagramı verilmiştir.



Şekil 4.16 SvmAddWindow fonksiyonu.

4.5 Temel Fonksiyonlar

Sistemin çalışmasını düzenleyen temel fonksiyonlardır.

4.5.1 SvmMain Fonksiyonu

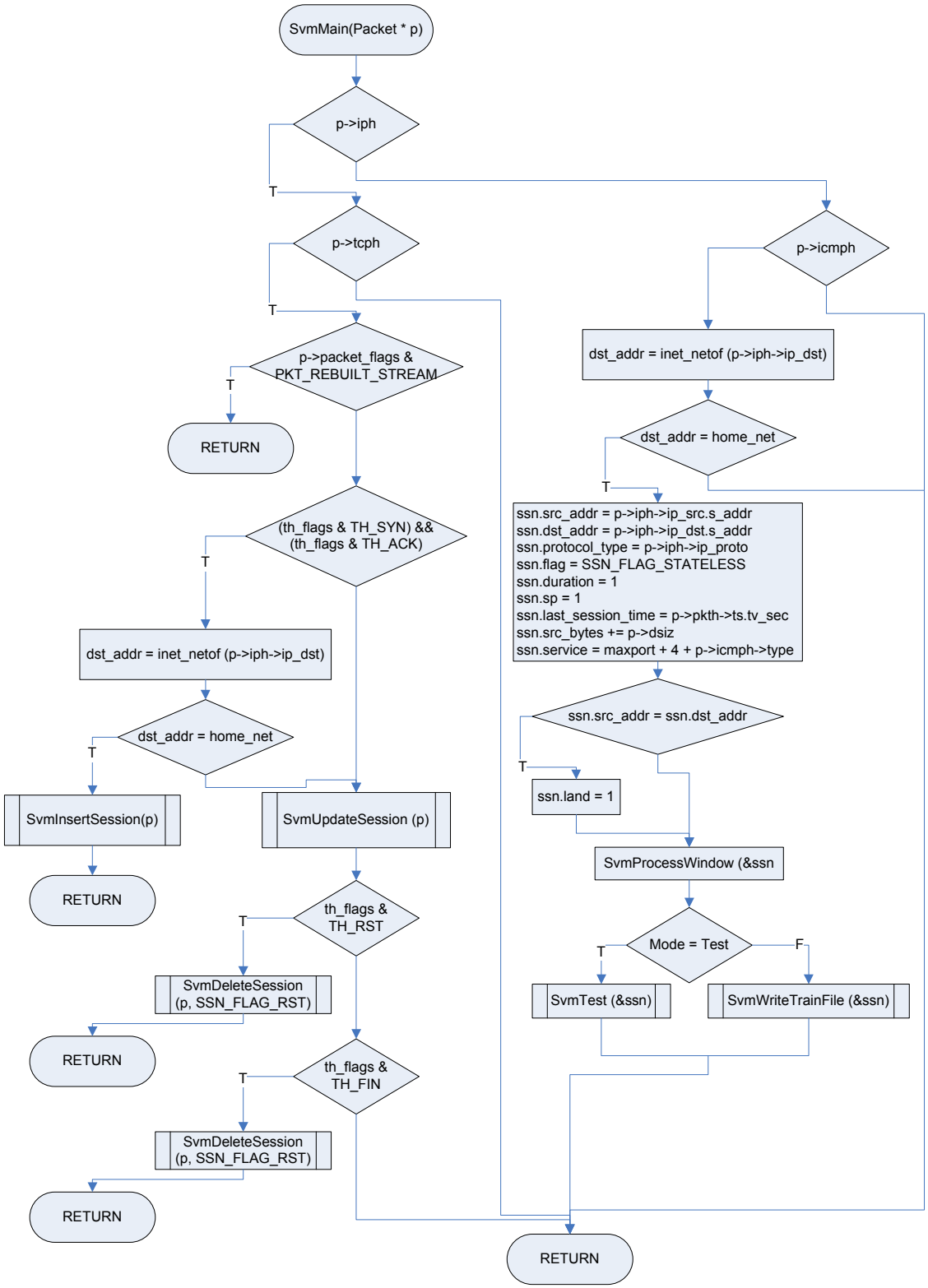
Sistemin ana fonksiyonudur. Her paket için Snort tarafından çağrılır. Gelen pakete ait bir işaretçi, fonksiyona parametre olarak gönderilir. SvmMain fonksiyonun işlevi, olabildiğince yalın bir şekilde gelen paketi incelemek ve uygun işlemlerden geçirmek üzere ilgili alt fonksiyonlara göndermektir. SvmMain fonksiyonu paketin başlık bilgisini kontrol ederek yapılacak işe karar verir. Yeni bir oturumun açılması için;

- Gelen paket TCP protokolünde olmalıdır.
- TCP başlık bilgisinde SYN ve ACK alanlarının değeri 1 olmalıdır.
- Bağlantı, dış ağdan iç ağa doğru geliyor olmalıdır.

Bir oturumun silinmesi için;

- Gelen paket TCP protokolünde olmalıdır.
- TCP başlık bilgisinde RST veya REJ alanlarının değeri 1 olmalıdır.
- Gelen pakete ait daha önce oluşturulmuş aktif halde bir oturum bilgisi bulunmalıdır.

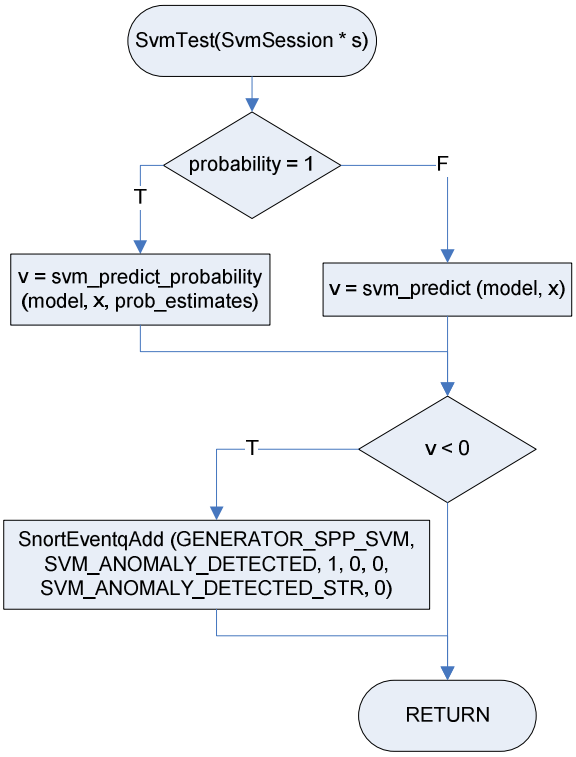
Fonksiyon, bağlantısız iletişim protokolüne ait bir ağ paketinin öznitelik değerlerini hesaplayarak arabelleğe ekler. SvmMain fonksiyonu sadece iç bilgisayar ağına doğru açılan trafiği inceler. Bir kere oturum açıldıktan sonra ise her iki yönlü trafik akışını tespit eder. Bu şekilde başarım ve doğruluk artışı sağlanmış olur. Sistem, TCP ve ICMP protokollerini desteklemektedir. Şekil 4.17'de fonksiyonun akış diyagramı verilmiştir.



Şekil 4.17 SvmMain fonksiyonu.

4.5.2 SvmTest Fonksiyonu

Fonksiyon, sınama kipinde çağrılmaktadır. Bir oturum veya paket hakkında gerekli öznitelikler toplandıktan sonra sınanmak üzere SvmTest fonksiyonuna iletilir. Burada, eğitim sonucu oluşturulmuş olan model ile kıyaslanır. Sınama sonucu, olasılıklı veya ikili olarak geri döndürülür. Sonuç değeri bir saldırıyı işaret ediyorsa Snort'un çıkış eklentisine bu olay iletilir. Şekil 4.18'de fonksiyonun akış diyagramı verilmiştir.



Şekil 4.18 SvmTest fonksiyonu.

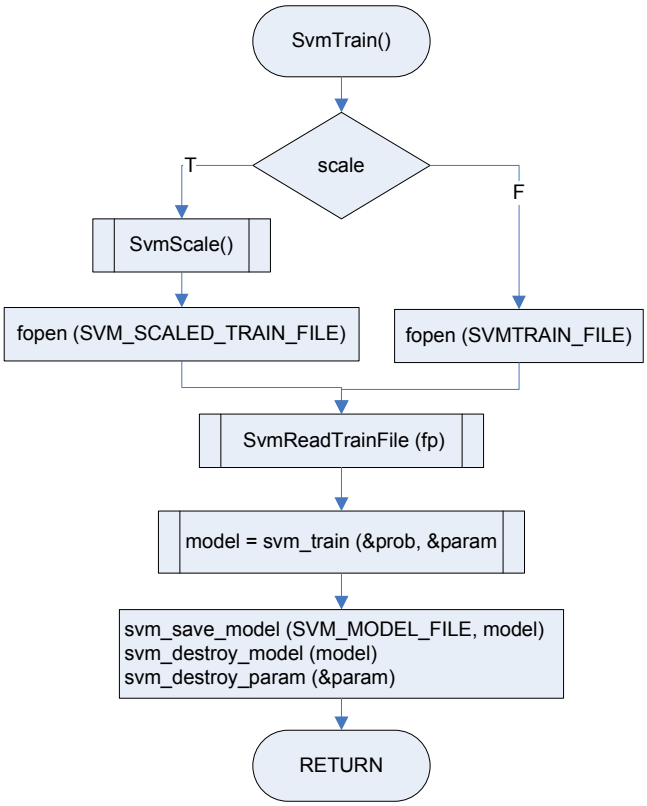
4.5.2.1 SnortEventqAdd Fonksiyonu

Snort'a ait olan bu fonksiyon, çıkış eklentilerine bir olay göndermek için kullanılır. Fonksiyona gönderilen ön tanımlı değerler (GENERATOR_SPP_SVM, SVM_ANOMALY_DETECTED, SVM_ANOMALY_DETECTED_STR) generators.h ve event_wrappers.h kütüphanelerinde tanımlanmıştır. Ön tanımlı değerler olayın hangi parça

tarafından üretildiğini, tipini ve mesaj iletisini içerir. Ön tanımlı ifadelerin kullanılması yapıyı standart hale getirmektedir.

4.5.3 SvmTrain Fonksiyonu

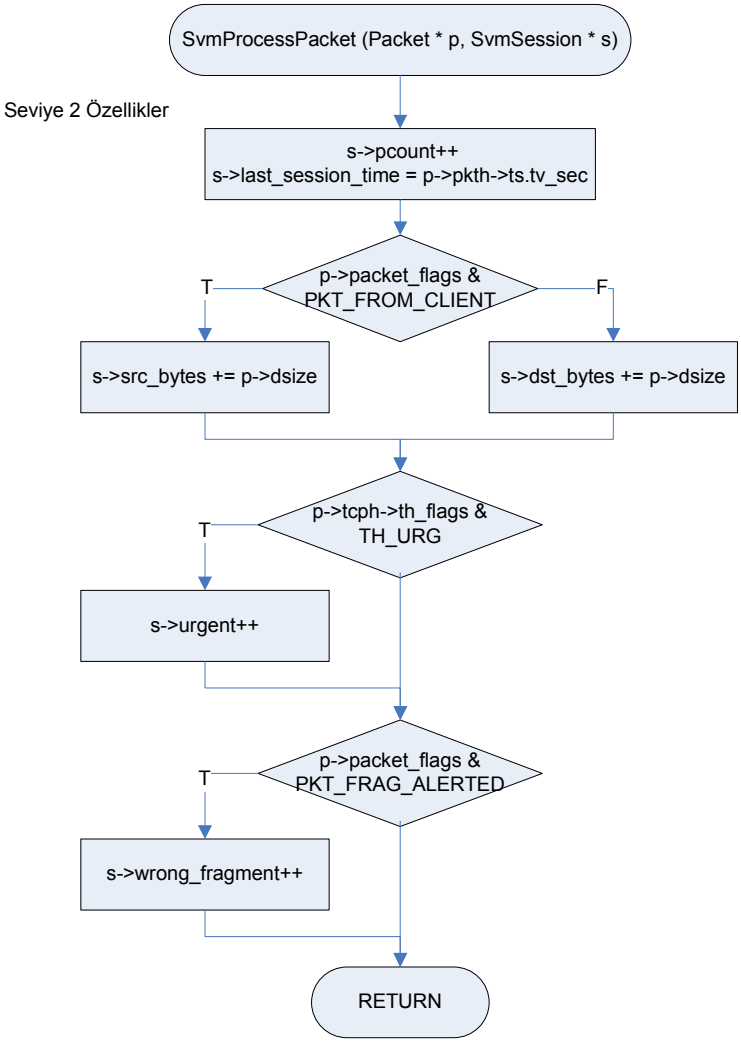
Trafik analizi aşamalarından elde edilen öznitelikleri kullanarak, sistemin eğitilmesini sağlar. Verinin ölçeklenmiş olup olmadığını kontrol eder. Sonucu model dosyasına kaydeder. Şekil 4.19'da fonksiyonun akış diyagramı verilmiştir.



Şekil 4.19 SvmTrain fonksiyonu.

4.5.4 SvmProcessPacket Fonksiyonu

Paketlerin, 2. seviye öznitelik değerlerini hesaplar. 2. seviye öznitelikler bir oturum boyunca değişim gösterirler ve gelen her paket için tekrar hesaplanırlar. Şekil 4.20'de fonksiyonun akış diyagramı verilmiştir.



Şekil 4.20 SvmProcessPacket fonksiyonu.

4.6 Diğer Fonksiyonlar

Sistemin genelinde kullanılan ve işleyişe yardımcı fonksiyonlardır.

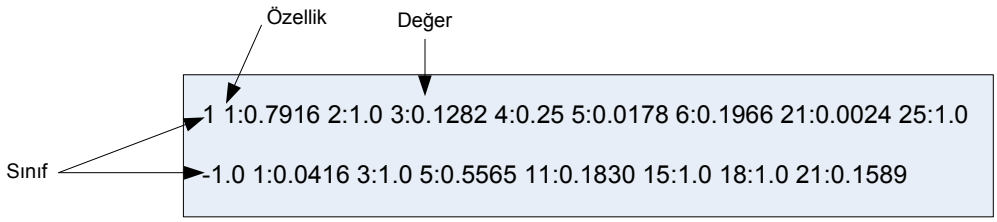
4.6.1 SvmScale Fonksiyonu

Analiz sonucu elde edilen özniteliklerin [0,1] değer aralığında ölçeklenmesini sağlar. Bu ölçekleme doğruluk oranını arttırmaktadır (Hsu et al., 2003). Ölçekleme oranları bir dosyaya

kaydedilmekte ve sınama aşamasında sınanacak öznitelikler öncelikle bu ölçek değerleriyle düzeltilmektedir.

4.6.2 Output Fonksiyonu

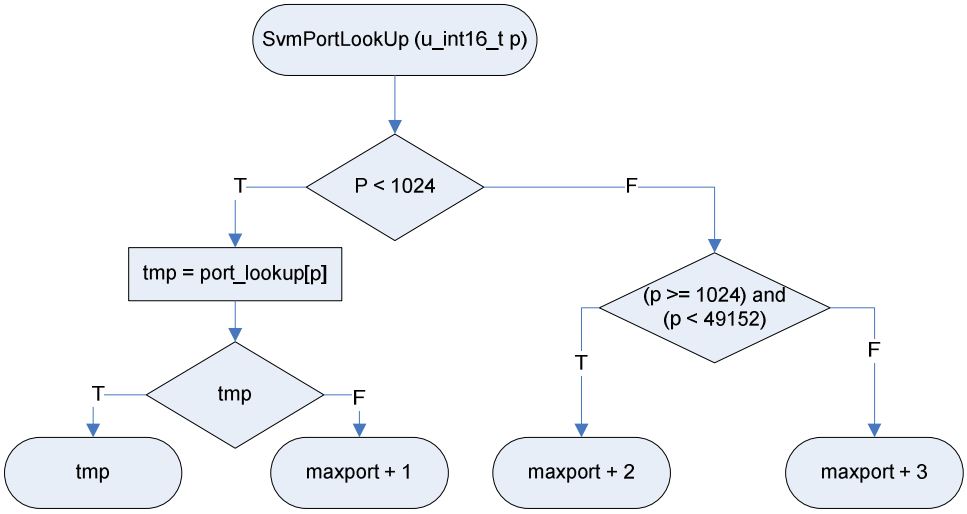
Öznitelik değerlerini libsvm dosya formatında kaydeder. Libsvm formatı Şekil 4.21'deki gibidir.



Şekil 4.21 Libsvm dosya formatı.

4.6.3 SvmPortLookUp Fonksiyonu

TCP/UDP port numaraları 0–65535 aralığında değişmektedir. Bilinen uygulamalar tarafından kullanılan port numaraları ise 0–1024 aralığındadır [19]. Normal bir ağ trafiğinde kullanılan ağ uygulamaları genelde sınırlıdır. Diğer port numaralarına açılan trafik genelde bir saldırı belirtisidir. Kullanılmayan büyük bir aralığı analiz etmek, kullanılan ve analiz için değeri olan bilgilerin kaybolmasına yol açmaktadır. Bu nedenle bir eşleşme tablosu yaratılarak sıklıkla kullanılan ağ uygulamaları bu tabloya eklenmiştir. SvmPortLookUp fonksiyonu verilen bir port numarasına karşılık port numarasının eşleştirildiği yeni değeri döndürür. Eğer port numarası eşleşme tablosunda bulunmuyorsa, ön tanımlı diğer 3 değerden uygun olanı ile eşleştirilir. Şekil 4.22'de fonksiyonun akış diyagramı verilmiştir.



Şekil 4.22 SvmPortLookUp fonksiyonu.

4.6.4 DebugMessage

Program hata ayıklama kipinde çalıştırıldığında, DEBUG_WRAP makrosunun içindeki ifadeler çalıştırılır. DebugMessage, mesajın geldiği parçanın ismini, satır numarasını ve hata mesajını ekrana yazar.

```
DEBUG_WRAP (DebugMessage (DEBUG_SVM, "%.16g %.16g\n", feature_max[i], feature_min[i])
```

4.6.5 FatalError

Snort'a ait bir akış kontrol fonksiyonudur. Ekrana parçaya ait isim, satır numarası ve hata mesajı bilgilerini yazar. Fonksiyon sonucunda Snort programının çalışması durdurulur. Sistemin çalışması içi gereken bir kaynağın var olup olmadığının kontrolünde kullanılır.

4.6.6 LogMessage, ErrorMessage

Bu iki fonksiyon konsola günlük bilgisi veya hata mesajı yazmak için kullanılır.

5. KULLANILAN VERİ YAPILARI

Sistemde kullanılan veri yapıları, kullanım amaçları, içerdikleri alanlar ve alanların amaçları aşağıda belirtilmiştir.

5.1 Snort Veri Yapıları

Snort tarafından tanımlanmış veri yapılarıdır. Fiziksel ortamdan alınan ağ paketi, çözümleme aşamasında kendisine uygun veri yapılarına eklenir ve daha sonraki işlemler bu veri yapıları üzerinden yapılır.

5.1.1 Packet Veri Yapısı

Snort'ta ağ paketi bilgisinin tutulduğu genel veri yapısıdır. Ağ paketine ait başlık işaretçileri, protokollere ait belirli özellikleri ve çeşitli ön işlemcilerin kullandığı veri yapıları işaretçilerini içerir. *Packet* veri yapısı paket çözümleme aşamasında oluşturulur. Daha sonra sırasıyla önişlemci ve tespit motoru eklentilerine iletilir. Ön işlemciler çalışmaları sonucu elde ettikleri sonuçları sonraki aşamalarda kullanılması için *packet* veri yapısına eklerler. Şekil 5.1'de *packet* veri yapısı gösterilmiştir.

```
typedef struct _Packet
{
    struct pcap_pkthdr *pkth;   /* BPF data */
    u_int8_t *pkt;              /* base pointer to the raw packet data */

    Fddi_hdr *fddihdr;          /* FDDI support headers */
    Fddi_llc_saps *fddisaps;
    Fddi_llc_sna *fddisna;
    Fddi_llc_iparp *fddiiparp;
    Fddi_llc_other *fddiother;

    Trh_hdr *trh;               /* Token Ring support headers */
    Trh_llc *trhllc;
    Trh_mr *trhmr;

    SLLHdr *sllh;               /* Linux cooked sockets header */

    PflogHdr *pfh;              /* OpenBSD pflog interface header */

    OldPflogHdr *opfh;          /* Old OpenBSD pflog interface header */

    EtherHdr *eh;               /* standard TCP/IP/Ethernet/ARP headers */
    VlanTagHdr *vh;
    EthLlc   *ehllc;
    EthLlcOther *ehllcother;

    WifiHdr *wifih;         /* wireless LAN header */

    EtherARP *ah;

    EtherEapol *eplh;       /* 802.1x EAPOL header */
    EAPHdr *eaph;
    u_int8_t *eaptype;
    EapolKey *eapolk;

    PPPoEHdr *pppoeh;       /* Encapsulated PPP of Ether header */

    IPHdr *iph, *orig_iph;
    u_int32_t ip_options_len;
    u_int8_t *ip_options_data;

    TCPHdr *tcph, *orig_tcph;
    u_int32_t tcp_options_len;
    u_int8_t *tcp_options_data;

    UDPHdr *udph, *orig_udph;
    ICMPHdr *icmph, *orig_icmph;

#ifdef GRE
    GREHdr *greh;
#endif

    u_int8_t *data;         /* packet payload pointer */
    u_int16_t dsize;        /* packet payload size */
    u_int16_t alt_dsize;    /* the dsize of a packet before munging (used for log)*/

    u_int16_t actual_ip_len;/* for logging truncated packets (usually by a small snaplen) */

    u_int8_t frag_flag;     /* flag to indicate a fragmented packet */
    u_int16_t frag_offset;  /* fragment offset number */
    u_int8_t mf;            /* more fragments flag */
    u_int8_t df;            /* don't fragment flag */
    u_int8_t rf;                /* IP reserved bit */

    u_int16_t sp;           /* source port (TCP/UDP) */
    u_int16_t dp;           /* dest port (TCP/UDP) */
    u_int16_t orig_sp;      /* source port (TCP/UDP) of original datagram */
    u_int16_t orig_dp;      /* dest port (TCP/UDP) of original datagram */
    u_int32_t caplen;

    u_int8_t uri_count;     /* number of URIs in this packet */

    void *ssnptr;           /* for tcp session tracking info... */
    void *fragtracker;      /* for ip fragmentation tracking info... */
    void *flow;             /* for flow info */
    void *streamptr;        /* for tcp pkt dump */

    Options ip_options[IP_OPTMAX]; /* ip options decode structure */
    u_int32_t ip_option_count;  /* number of options in this packet */
    u_char ip_lastopt_bad;  /* flag to indicate that option decoding was
                               halted due to a bad option */
    Options tcp_options[TCP_OPTLENMAX];    /* tcp options decode struct */
    u_int32_t tcp_option_count;
    u_char tcp_lastopt_bad;  /* flag to indicate that option decoding was
                                halted due to a bad option */

    u_int8_t csum_flags;        /* checksum flags */
    u_int32_t packet_flags;     /* special flags for the packet */
    u_int32_t bytes_to_inspect; /* Number of bytes to check against rules */

    BITOP *preprocessor_bits;  /* flags for preprocessors to check */
} Packet;
```

Şekil 5.1 Packet veri yapısı.

5.1.2 IPHdr Veri Yapısı

IP protokol başlık bilgisinin tutulduğu veri yapısıdır. Şekil 5.2'de IP başlık yapısı verilmiştir.

```
typedef struct _IPHdr
{
    u_int8_t ip_verhl;      /* version & header length */
    u_int8_t ip_tos;        /* type of service */
    u_int16_t ip_len;       /* datagram length */
    u_int16_t ip_id;        /* identification  */
    u_int16_t ip_off;       /* fragment offset */
    u_int8_t ip_ttl;        /* time to live field */
    u_int8_t ip_proto;      /* datagram protocol */
    u_int16_t ip_csum;      /* checksum */
    struct in_addr ip_src;  /* source IP */
    struct in_addr ip_dst;  /* dest IP */
}     IPHdr;
```

Şekil 5.2 IP başlığı.

5.1.3 TCPHdr Veri Yapısı

TCP protokol başlık bilgisinin tutulduğu veri yapısıdır. Şekil 5.3'de TCP başlık yapısı verilmiştir.

```
typedef struct _TCPHdr
{
    u_int16_t th_sport;     /* source port */
    u_int16_t th_dport;     /* destination port */
    u_int32_t th_seq;       /* sequence number */
    u_int32_t th_ack;       /* acknowledgement number */
    u_int8_t th_offx2;      /* offset and reserved */
    u_int8_t th_flags;
    u_int16_t th_win;       /* window */
    u_int16_t th_sum;       /* checksum */
    u_int16_t th_urp;       /* urgent pointer */

}     TCPHdr;
```

Şekil 5.3 TCP başlığı.

5.2 Yerel Veri Yapıları

Tasarlanan sistem tarafından kullanılan yerel veri yapılarıdır.

5.2.1 SvmGlobalConfig Veri Yapısı

Sistem genelinde kullanılan ön tanımlı değerleri saklayan veri yapısıdır. Eğer varsa, Snort.conf dosyasındaki değerleri, yoksa ise varsayılan değerleri içerir. Şekil 5.4'de veri yapısı gösterilmiştir.

```
typedef struct {

    int8_t mode;
    u_int8_t scale;
    u_int16_t max_conn;
    u_int16_t timeout;
    u_int16_t win_size;
    u_int32_t home_net;
    time_t win_time;
} SvmGlobalConfig;
```

Şekil 5.4 SvmGlobalConfig veri yapısı.

5.2.2 SvmSession Veri Yapısı

Oturuma ait durum bilgisinin ve öznitelik değişkenlerinin saklandığı veri yapısıdır. Şekil 5.5'de veri yapısı gösterilmiştir.

```
typedef struct {

   u_int32_t pcount;
   SvmSessionKey hashKey;          // Hash Table Key
   struct timeval start_time;      /* unix second the session started */
   time_t last_session_time;       /* last time this session got a packet */
   u_int32_t src_addr;             // Source address
   u_int16_t sp;                   // Source port
   u_int32_t dst_addr;             // Destination address
   u_int16_t dp;                   // Destination port
   u_int8_t serror;                // SYN error

   u_int32_t duration;             // length (# of seconds) of the connection
   u_int8_t protocol_type;         // Type of the protocol, e.g. tcp, udp, icmp
   u_int16_t service;              // Network services on the destination, e.g. http, telnet, etc.
   u_int8_t flag;                  // Normal or error status of the connection
   u_int32_t src_bytes;            // # of data bytes from source to destination
   u_int32_t dst_bytes;            // # of data bytes from destination to source
   u_int8_t land;                  // 1 if connection is from/to the same host/port; 0 otherwise
   u_int16_t wrong_fragment;       // # of "wrong" fragments
   u_int16_t urgent;               // # of urgent packets

   u_int16_t count;
   u_int16_t srv_count;
   u_int8_t serror_rate;
   u_int8_t srv_serror_rate;
   u_int8_t rerror_rate;
   u_int8_t srv_rerror_rate;
   u_int8_t same_srv_rate;
   u_int8_t diff_srv_rate;
   u_int8_t srv_diff_host_rate;
   u_int8_t same_src_port_rate;    //extra

   u_int16_t dst_host_count;
   u_int16_t dst_host_srv_count;
   u_int8_t dst_host_same_srv_rate;
   u_int8_t dst_host_diff_srv_rate;
   u_int8_t dst_host_same_src_port_rate;
   u_int8_t dst_host_srv_diff_host_rate;
   u_int8_t dst_host_serror_rate;
   u_int8_t dst_host_srv_serror_rate;
   u_int8_t dst_host_rerror_rate;
   u_int8_t dst_host_srv_rerror_rate;
} SvmSession;
```

Şekil 5.5 SvmSession veri yapısı.

5.2.3 SvmSessionKey Veri Yapısı

Kıyım anahtarının saklandığı veri yapısıdır. Veri yapısındaki değişkenler her oturum için eşsiz bir anahtar değeri üretmektedirler. Şekil 5.6'da veri yapısı gösterilmiştir.

```
typedef struct _sessionkey {
    u_int32_t lowIP;
    u_int32_t highIP;
    u_int16_t port;
    u_int16_t port2;
    u_int8_t proto;
} SvmSessionKey;
```

Şekil 5.6 SvmSessionKey veri yapısı.

5.2.4 SvmWindow Veri Yapısı

Geçmiş oturumlara ait durum bilgisini tutan veri yapısıdır. Sistemde kullanılan arabellek yapısını oluşturur. Şekil 5.7'de veri yapısı gösterilmiştir.

```
typedef struct {

    u_int32_t ip_addr;
    u_int16_t service;
    u_int16_t sp;
    u_int8_t syn_err;
    u_int8_t rej_err;
    time_t time;
} SvmWindow;
```

Şekil 5.7 SvmWindow veri yapısı.

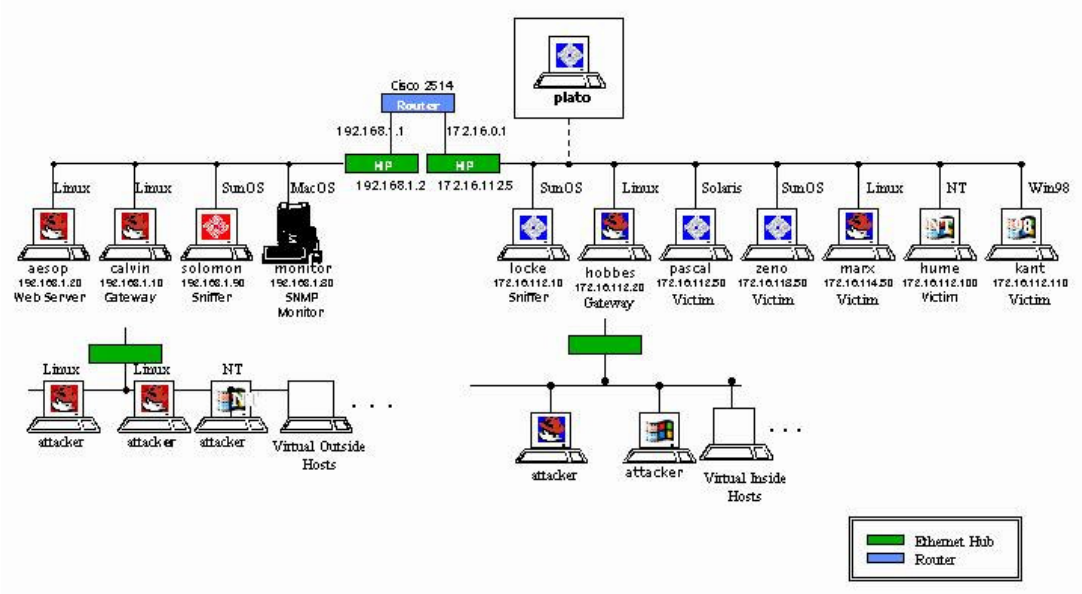
6. SINAMA

Sistemin sınanması için DARPA (Defense Advanced Research Project Agency) [14] tarafından saldırı tespit sistemlerinin değerlendirmesi amacıyla üretilen veri kümesi kullanılmıştır. Sistemin, farklı giriş parametreleri için tanıma başarısı ölçülmüştür. Tanıma için kullanılan öznitelik değerleri farklı alt kümeler halinde sınamaya tabii tutulmuş, tanımaya etki eden en iyi öznitelik değerleri araştırılmıştır. Destek vektör makinesinin başarımının arttırılması için çeşitli sınamalar yapılmıştır. Sistemin ürettiği veriler alınarak farklı sınıflandırma algoritmalarına verilmiş ve sonuçlar karşılaştırılmıştır. Sistem, diğer saldırı tespit sistemleriyle karşılaştırılmıştır. Sonuçlar, farklı başarım metriklerine göre değerlendirilmiştir.

Sınama aşamalarının önemli bir kısmında Weka sınıflandırma aracından [28] yararlanılmıştır. Weka, Java programlama dilinde yazılmış olan, birçok sınıflandırma ve öğrenme algoritmaları içeren bir araçtır. Weka'da bulunan standart algoritmalar dışında Weka için yazılan bazı eklentilerden de yararlanılmıştır. SOM algoritması için *Wekaclassalgos* eklentisi [29], SVM algoritması için *Wlsvm* eklentisi [30] kullanılmıştır.

6.1 DARPA Veri Kümesi

İlk defa 1998 yılında, DARPA ve ABD Hava Kuvvetleri Araştırma Laboratuvarı'nın [20] desteğiyle, MIT (Massachusetts Institute of Technology) Lincoln Laboratuarı Bilişim Sistemleri Teknoloji Grubu tarafından saldırı tespit sistemlerini değerlendirmek amacıyla oluşturulmuştur [14]. 1999 yılında, veri kümesi daha geliştirilerek ve yeni atak bilgileri eklenerek tekrar üretilmiştir. Tez çalışmasının sınanmasında 1999 yılına ait veri kümesi kullanılmıştır. DARPA tarafından oluşturulan ortam, ABD Hava Kuvvetleri'ne ait bir yerel bilgisayar ağının benzetimini yapmaktadır. Şekil 6.1'de benzetim yapılan ağın şeması verilmiştir. DARPA verisi, 3 hafta boyunca yerel ağda oluşturulan trafiğin TCP/IP döküm bilgisinden oluşmaktadır. 1. ve 3. haftalar saldırı içermeyen "temiz" trafikten oluşmaktadır. 2. haftada ise saldırı trafiği içermektedir.



Şekil 6.1 Benzetim yapılan ağın şeması. [21]

Benzetimde kullanılan saldırılar 4 ana kategoride toplanabilir.

1. Hizmet engelleme saldırıları
2. Uzak bilgisayardan yapılan saldırılar
3. Yetki kazanma saldırıları
4. Araştırma ve gözlem

6.1.1 Hizmet Engelleme Saldırıları

Hizmet engelleme saldırıları, hedef bilgisayarın hafıza, ağ ve işlemci kaynaklarını sömürerek diğer isteklere cevap vermesini engeller.

6.1.2 Uzak Bilgisayardan Yapılan Saldırılar

Uzak bilgisayardan yapılan saldırılarda, saldırganın hedef bilgisayar üzerinde herhangi bir kullanıcı hesabı bulunmamaktadır. Amacı, hedef bilgisayarda bir kullanıcı hesabı elde etmektir.

6.1.3 Yetki Kazanma Saldırıları

Yetki kazanma saldırılarında saldırganın hedef bilgisayar üzerinde bir kullanıcı hesabı bulunmaktadır. Saldırganın amacı kendi kullanıcısının yetki seviyesinden daha yetkili bir seviyeye geçmektir (Örn.: root kullanıcısı).

6.1.4 Araştırma ve Gözlem

Araştırma ve gözlem saldırıları pasif saldırılardır. Daha önce belirtilen saldırı tipleri kullanılmadan önce araştırma ve gözlem yapılarak hedef bilgisayar hakkında daha ayrıntılı bilgi edinilmesi amaçlanmaktadır.

DARPA veri kümesinde yukarıda belirtilen 4 ana kategoride toplam 32 farklı saldırı kullanılmıştır. Bu saldırılar ve ait oldukları kategoriler Çizelge 6.1'de gösterilmiştir.

Çizelge 6.1 DARPA veri kümesinde kullanılan saldırılar [22]

Hizmet Aksatma Saldırıları	**Yetki Kazanma Saldırıları**	**Uzaktan Yapılan Saldırılar**	**Araştırma ve Gözlem**
Apache2	anypw*	Dictionary	insidesniffer*
arppoison*	casesen*	Ftpwrite	Ipsweep
Back	Eject	Guest	ls_domain*
Crashiis	Ffbconfig	Httptunnel	Mscan
dosnuke*	Fdformat	Imap	NTinfoscan
Land	Loadmodule	Named	Nmap
Mailbomb	ntfsdos*	ncftp*	queso*
SYN Flood(Neptune)	Perl	netbus*	resetscan*
Ping of Death (POD)	Ps	netcat*	Saint
Process Table	sechole*	Phf	Satan
selfping*	Xterm	ppmacro*	
Smurf	yaga*	Sendmail	
sshprocesstable*		sshtrojan*	
Syslogd		Xlock	
tcpreset*		Xsnoop	
Teardrop			
Udpstorm			

Çizelge 6.2'de DARPA benzetiminde Pazartesi günü boyunca yapılan saldırıların başlangıç saati, hedefi ve saldırıların isimleri verilmiştir. Saldırı sayısı ve saatinden de anlaşılacağı

üzere tüm gün yapılan benzetimde toplam 7 adet saldırı gerçekleştirilmiştir. Geri kalan trafik normal bilgisayar ağı trafiğidir. Saldırıların seyrek ve zamana yayılmış olması daha gerçekçi bir sınama için önemlidir.

Çizelge 6.2 DARPA benzetiminde bir gün içerisinde yapılan saldırılar [14].

Sıra No	Tarih	Başlangıç Zamanı	Hedef	Saldırı
1	03.08.1999	08:01:01	hume.eyrie.af.mil	NTinfoscan
2	03.08.1999	08:50:15	zeno.eyrie.af.mil	pod
3	03.08.1999	09:39:16	marx.eyrie.af.mil	back
4	03.08.1999	12:09:18	pascal.eyrie.af.mil	httptunnel
5	03.08.1999	15:57:15	pascal.eyrie.af.mil	land
6	03.08.1999	17:27:13	marx.eyrie.af.mil	secret
7	03.08.1999	19:09:17	pascal.eyrie.af.mil	ps attack

6.2 KDD Cup Veri Kümesi

1999 yılındaki 3. Uluslararası Bilgi Keşfi ve Veri Madenciliği Araçları Yarışması'nda (KDD Cup'99) [13] anormallik tespiti ile tanıma yapan ağ saldırı tespit sistemlerinin başarı oranlarını ölçmek için oluşturulmuştur. KDD Cup veri kümesi DARPA veri kümesinden oluşturulmuş olup her bilgisayar ağı oturumu için 41 adet öznitelik içermektedir. Birçok bilimsel çalışmada, anormallik tespiti algoritmalarının başarımını ölçmek için KDD Cup veri kümesi kullanılmaktadır.

6.3 Başarımın Ölçülmesi

Sistemin değerlendirilmesinde eğitim ve sınama olmak üzere iki farklı veri kümesi kullanılmıştır. Başarımın doğru ölçülebilmesi için bu iki veri kümesinde kullanılan örneklerin birbirlerinden farklı olmaları gerekmektedir. Tek seferde yapılan sınamalar da doğruluğun

tespiti için yeterli olmazlar. Sınıflandırma algoritmalarının başarımlarını doğru tespit etmek için aşağıda belirtilen sınama yöntemleri kullanılmaktadır (Kohavi, 1995).

- Çapraz Geçerlilik Sınaması (Cross Validation)
- Önyükleyici Kurgulaması (Bootstrap Aggregating (Bagging))

6.3.1 Çapraz Geçerlilik Sınaması

Verinin az olduğu durumlarda sistemin sınanması için kullanılır. N adet veri k adet alt kümeye bölünür. Her seferde 1 küme eğitim kümesi olurken k-1 adeti de sınama kümesi olur. K adet eğitim ve sınama yapıldıktan sonra sonuçların ortalaması alınır. Sistemin sınanmasında k değeri 10 olarak alınmıştır.

6.3.2 Önyükleyici Kurgulaması

Eğitim kümesi için veri kümesinden örnekler seçilir. Ancak, seçilen örnek tekrar veri kümesine geri konur. Bu nedenle, eğitim kümesinde veri kümesinden seçilen aynı örnekler bulunabilir. İşlem eğitim kümesindeki eleman sayısı istenen değere (n) ulaşıncaya kadar devam eder. Genellikle n değeri veri kümesindeki örnek sayısı kadardır. Sınama kümesi, eğitim kümesi için seçilen örneklerin dışında oluşturulur. Bu yöntem daha büyük eğitim kümeleri elde etmek için kullanılır.

Sınama için kullanılan DARPA veri kümesi yeterince büyük olduğu için veri kümesi iki parçaya ayrılarak kullanılmıştır. Veri kümesi, rasgele seçilen verilerden oluşturulmuş, %10'u eğitim ve %90'ı sınama için ayrılmıştır. Kararlılığın önemli olduğu durumlarda ise çapraz geçerlilik sınaması yapılmıştır.

6.3.3 Başarım Metrikleri

Bir sınıflandırma algoritmasının başarımını ölçmek için başlıca 6 farklı metrik kullanılmaktadır..

- Doğruluk (Accuracy)
- Kesinlik (Precision)
- Geri Çağırım (Recall)
- F-Ölçütü (F-Measure)
- Geometrik Ortalama1 (G-mean1)
- Geometrik Ortalama2 (G-mean2)

Yukarıda belirtilen metrikler Çizelge 6.3'deki hata matrisindeki değerler yardımıyla hesaplanmaktadır. Pozitif olarak ölçülen değer gerçekte de pozitif sınıfta ise sonuç doğru pozitiftir (DP). Negatif olarak ölçülen değer gerçekte de negatif sınıfa ait ise sonuç doğru negatiftir (DN). Pozitif olarak ölçülen bir değer gerçekte negatif sınıfa ait ise sonuç yanlış pozitiftir (YP). Negatif olarak ölçülen bir değer gerçekte pozitif sınıfa aitse sonuç yanlış negatiftir (YN). Metriklerin formülleri Formül 9.1'de verilmiştir.

Çizelge 6.3 Hata matrisi.

		Gerçek	
		Sınıf$_+$	Sınıf$_-$
Tahmin Edilen	Sınıf$_+$	Doğru Pozitif (DP)	Yanlış Pozitif (YP)
	Sınıf$_-$	Yanlış Negatif (YN)	Doğru Negatif (DN)

$$
\begin{aligned}
&Dogruluk = \frac{DP + DN}{DP + DN + YP + YN} \\
&Ke\sin lik(K) = \frac{DP}{DP + YP} \\
&GeriÇagirim(G) = \frac{DP}{DP + YN} \\
&F - Ölçütü = \frac{1}{F} = \frac{1}{2}\left(\frac{1}{K} + \frac{1}{G}\right), \\
&F = \frac{2KG}{K + G} \\
&DNO = \frac{DN}{DN + YP} \\
&G - Ortalama1 = \sqrt{G * K} \\
&G - Ortalama2 = \sqrt{G * DNO}
\end{aligned}
\qquad (9.1)
$$

Sınıflandırma algoritmalarının başarımını ölçmek için genelde *doğruluk* metriği kullanılmaktadır. Ancak, sınanan bir sınıfın diğer sınıf(lar)a göre daha az sayıda örneği olduğu durumlarda *doğruluk* metriği zayıf kalır. *Kesinlik* ve *geri çağırım* metrikleri bu durumda daha iyi bir ölçüm sağlarlar. *F-ölçütü*, *kesinlik* ve *geri çağırım* metriklerinin bir ortalamasıdır (Musicant et al., 2003). *Kesinlik*, pozitif olarak tahmin edilen değerler içindeki doğru tahminlerin oranıdır. *Geri Çağırım*, doğru pozitif olarak ölçülen değerlerin tüm gerçek pozitif sınıfa oranıdır. *Doğru Negatif Oranı (DNO),* negatif olarak ölçülen değerlerin tüm gerçek negatif sınıfa oranıdır.

6.4 Öznitelik Başarımlarının Arttırılması

Sistem, 1999 yılındaki KDD (Knowledge Discovery and Data Mining) yarışmasında kullanılan 23 adet (Kullanılmayan 18 öznitelik sunucu seviyesinde bilgi gerektirmektedir.) öznitelik değerini ağ trafiğinden çıkarmaktadır. Çıkarılan her bir öznitelik değerinin tanımaya etkisi farklıdır. Öznitelik değerlerinin tanımaya etkilerinin belirlenmesi doğruluk oranı için önemlidir. Daha az ama tanımaya etkili özniteliklerin kullanımı hem doğruluk oranını hem de sistemin çalışma hızını arttırmaktadır.

Mukkamala, kullanılan veriyi azaltmak için her seferinde bir özniteliği dışarıda tutarak kalan 40 adet öznitelik için sınamalar yapmıştır. Böylece, dışarıda bırakılan özniteliğin tanımaya olan etkisi ölçülmüştür. Sonuç olarak, 41 adet öznitelikten Çizelge 6.4'te sol sütunda gösterilen 13 tanesini tanıma için belirgin olduğunu belirlemiştir (Mukkamala et al., 2002). Mukkamala, sınama için KDD Cup veri kümesini kullanmıştır. Karşılaştırma yapmak için aynı veri seti ile Weka sınıflandırma aracındaki öznitelik seçimi algoritmaları kullanılarak sınama yapılmıştır. Öznitelik seçimi için *CfsSubsetEval*, arama metodu için *BestFirst* algoritmaları kullanılmıştır. *CfsSubsetEval* algoritması öznitelikleri alt kümelere ayırır ve her bir alt küme için öngörme kabiliyeti ve artıklık derecesi dikkate alınarak değerlerini hesaplar. Yapılan sınama sonucunda algoritma, Çizelge 6.4'te sağ sütunda gösterilen 14 adet özniteliği seçmiştir. Koyu renk ile belirtilenler ortak özniteliklerdir.

Çizelge 6.4 KDD Cup veri kümesinin öznitelik alt kümeleri.

Öznitelikler (Mukkamala)	**Öznitelikler (BestFirst + CfsSubsetEval)**
duration	**land**
protocol_type	**protocol_type**
service	**service**
src_bytes	**src_bytes**
dst_bytes	**dst_bytes**
urgent	**logged_in**
count	**count**
srv_count	**srv_count**
same_srv_rate	diff_srv_rate
dst_host_count	root_shell
dst_host_srv_count	flag
dst_host_same_srv_rate	wrong_fragment
dst_host_same_src_port_rate	dst_host_same_src_port_rate
	serror_rate

Öznitelik alt kümelerinin başarımını ölçmek için 3 farklı veri seti oluşturulmuştur. Mukkamala'nın ve Weka yazılımının belirlediği özniteliklerden oluşan iki küme ve tüm öznitelikleri barındıran bir küme Svm algoritmasına verilmiştir. Verilerin 25274 adeti (%10'u) eğitimde, geri kalan 227467 adeti (%90'ı) sınamada kullanılmıştır.

Çizelge 6.5 KDD Cup verisindeki özniteliklerin azaltılması.

	41 Öznitelikli Küme	**CfsSubsetEval (14 Öznitelikli)**	**Mukkamala (13 Öznitelikli)**
Doğruluk Oranı	**99.51**	**98.72**	98.68
Eğitim Zamanı (CPU)	**12.99**	**14.38**	18.92
Sınama Zamanı (CPU)	**81.90**	**80.15**	100.32
Yanlış Sınıflandırılma	**1109**	2921	3003
Yanlış Negatifler	190	**105**	201
Yanlış Pozitifler	**919**	2816	2802

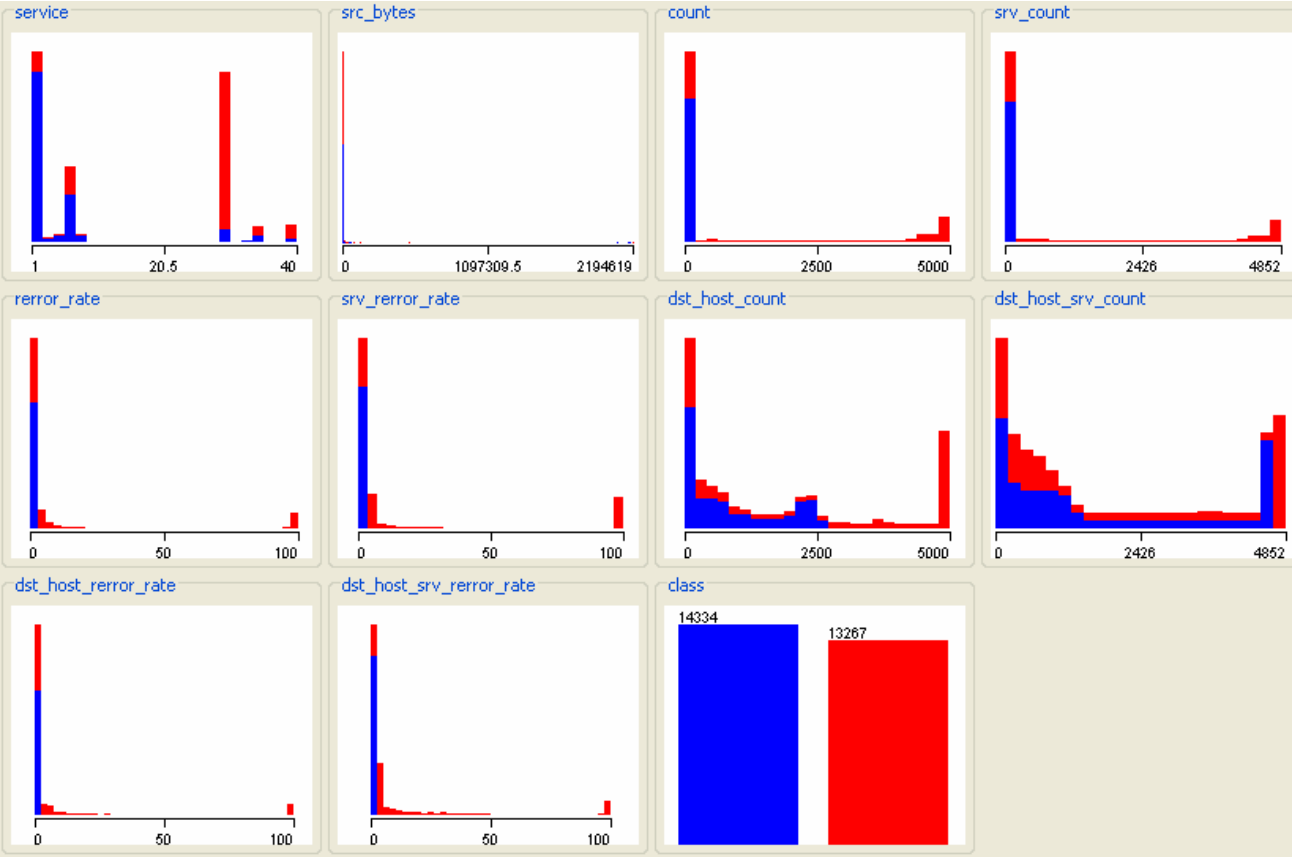
Çizelge 6.5'teki sınama sonucuna bakıldığında, doğruluk oranlarının birbirine çok yakın olduğu görülmektedir. CFSSubsetEval algoritması ile seçilen özniteliklerin Mukkamala'nın kullandığı özniteliklerden az da olsa daha başarılıdır. İşlem zamanlarına göre ise CFSSubsetEval ile seçilen özniteliklerin SVM algoritmasında daha hızlı sonuç ürettikleri görülmektedir.

Sistemin ürettiği öznitelik değerleri için de öznitelik değerlendirme sınaması yapılmıştır. Mukkamala'dan farklı olarak, öznitelikler teker teker çapraz geçerlilik sınamasına sokulmuş (k=10 alınmıştır) ve sonuçlar yanlış pozitif oranlarına göre küçükten büyüğe sıralanmıştır. Çizelge 6.6'te sonuçlar sıralı olarak listelenmiştir.

Çizelge 6.6 Sisteme ait özniteliklerin seçimi.

Öznitelik	Yanlış Pozitif Oranı
srv_rerror_rate	**0.22**
rerror_rate	**0.245**
dst_host_rerror_rate	**0.262**
dst_host_srv_rerror_rate	**0.327**
count	**0.54**
dst_host_srv_count	**0.546**
service	**0.608**
dst_host_count	**0.624**
srv_count	**0.625**
src_bytes	**0.632**
flag	0.643
dst_host_diff_srv_rate	0.649
duration	0.662
dst_host_same_srv_rate	0.665
dst_bytes	0.667
dst_host_srv_diff_host_rate	0.692
srv_diff_host_rate	0.77
diff_srv_rate	0.79
same_srv_rate	0.79
protocol_type	0.853
dst_host_same_src_port_rate	0.856
same_src_port_rate	0.857
urgent	1.0

İlk on özelliğin sınıflara göre dağılımı Şekil 6.2'de grafiksel olarak gösterilmektedir. Dağılımın grafiği Weka sınıflandırma aracı tarafından alınmıştır.



Şekil 6.2 Özniteliklerin sınıflara göre dağılımı.

Sistem tarafından öznitelik değerleri Weka sınıflandırma aracındaki öznitelik seçimi algoritmaları kullanılarak bir kez daha sınanmıştır. Yapılan sınama sonucunda algoritma, 4 adet özniteliği en belirleyici öznitelik olarak hesaplamıştır.

- protocol_type
- srv_rerror_rate
- dst_host_rerror_rate
- dst_host_srv_rerror_rate

23,10 ve 4 öznitelikli veri kümeleri çapraz geçerlilik sınaması (k=10 alınmıştır) ile SVM algoritmasından geçirilmiş ve sonuçları Çizelge 6.7'te gösterilmiştir. 23 öznitelikli küme yaklaşık %1 oranında daha iyi başarım gösterirken, 4 öznitelikli kümenin eğitim ve sınama zamanları daha iyidir. Gerçeklenen sistem gerçek zamanlı olması nedeni ile eğitim ve sınama

zamanları çok önemlidir. %1'lik başarım farkı, daha hızlı eğitim ve sınama zamanı için göz ardı edilebilir.

Çizelge 6.7 Seçilen özniteliklerin başarıma etkisi.

	23 Öznitelikli Küme	**10 Öznitelikli Küme**	**4 Öznitelikli Küme**
Doğruluk Oranı	**96.69**	95.68	95.60
Yanlış-Pozitif Oranı	**0.06**	0.09	0.09
F-Ölçütü	**0.97**	**0.96**	**0.96**
Eğitim Zamanı (CPU)	78.15	55.00	**17.39**
Sınama Zamanı (CPU)	6.02	4.37	**1.34**

6.5 Destek Vektör Makinesinin Başarımının Arttırılması

Destek vektör makinesinin tanıma başarımını arttırmak için aşağıda belirtilen konularda sistem sınanmıştır.

- Uygun çekirdek fonksiyonunun bulunması
- Parametre seçimi
- Verinin ölçeklenmesi

6.5.1 Uygun Çekirdek Fonksiyonunun Bulunması

Destek vektör makinesinde çekirdek tabanlı bir öğrenme algoritmasıdır. Algoritmanın uygulandığı probleme uygun bir çekirdek fonksiyonunun seçilmesi doğruluk oranını etkileyen önemli bir faktördür. 4 farklı çekirdek fonksiyonu için sınamalar yapılmıştır. Sınama, 22080 adet örnek için çapraz geçerlilik yöntemi ile yapılmıştır (k=10 alınmıştır). Çizelge 6.8'de sınama sonuçları verilmiştir.

Çizelge 6.8 Çekirdek fonksiyonlarının karşılaştırılması.

	Doğrusal	**Polinomsal**	**Radyal**	**Sigmoid**
Doğruluk Oranı	93.79	**96.47**	**96.68**	68.53
Yanlış-Pozitif Oranı	0.12	0.07	**0.06**	0.33
F-Ölçütü	0.94	**0.97**	**0.97**	0.70
Eğitim Zamanı (CPU)	319.65	245.94	**88.69**	198.30
Sınama Zamanı (CPU)	5.47	4.83	**3.49**	16.70

Sınama sonucunda radyal tabanlı çekirdek fonksiyonunun hem başarı hem de hız açısından en uygun seçim olduğu görülmüştür.

6.5.2 Parametre Seçimi

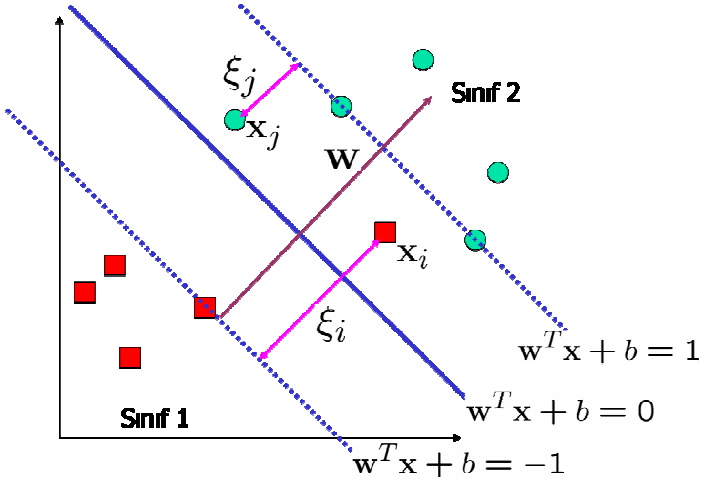
Destek vektör makinesinin sınıflandırma başarısı çekirdek fonksiyonu dışında verilen iki adet parametre ile de doğrudan ilişkilidir.

6.5.2.1 Maliyet çarpanı (C)

C katsayısı, SVM hata sayısı ile aşırı düzlemin en üst düzeye çıkarılması arasında bir denge sağlar [23].

6.5.2.2 Epsilon (ε)

Epsilon değeri, aşırı düzlemin belirlenmesi esnasında gürültüleri yok etmek amacıyla kullanılır. Sınıf sınırlarının daha yumuşak veya sert olmasını sağlar. Kullanılan veri kümesine göre verilecek değer değişiklik gösterir [23].

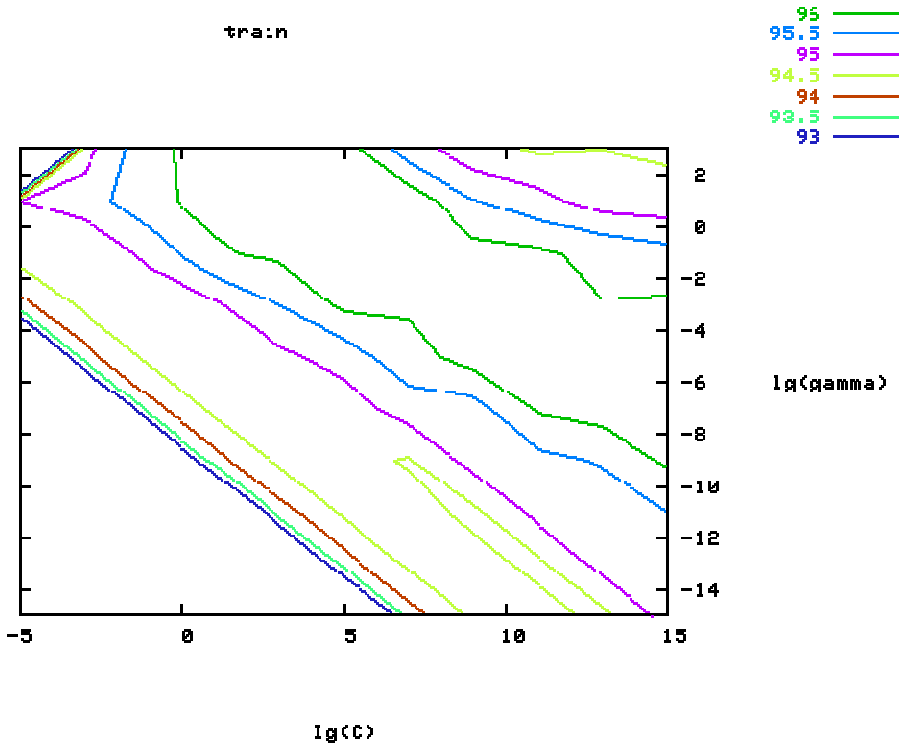


Şekil 6.3 Epsilon değeri.

Uygun parametrelerin seçimi bir NP problemdir. En uygun ikiliyi bulmak için belirli bir aralıkta sürekli sınamalar yapmak gerekir. Bu sınamalar için Libsvm içerisinde bulunan bir python betiğinden yararlanılmıştır. Betik, farklı C ve ε değerleri için svm kütüphanesini çağırmaktadır. Parametre seçimi paralel çalıştırılmaya uygun olduğu için sınama işlemi Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü bünyesinde bulunan TuxNet kümesinde yapılmıştır. TuxNet kümesi, IBM S/390 sistemi üzerinde yapılandırılmış 8 adet eşdeğer Linux (Çekirdek sürümü: 2.6.5-7.257) makineden oluşmaktadır. Şekil 6.4'te örnek konsol çıktısı verilmiştir. TuxNet üzerinde yapılan sınama sonuçları Şekil 6.5'te, betik tarafından üretilen grafikte gösterilmiştir. Grafikte parametrelerin iki tabanında logaritmik değerleri bulunmaktadır.

```
erdem@tux01] 5 -7 94.7464  (best c=32.0, g=0.0078125, rate=94.7464)
[erdem@tux04] -1 -1 95.1087  (best c=0.5, g=0.5, rate=95.1087)
[erdem@tux03] 5 -1 96.2681  (best c=32.0, g=0.5, rate=96.2681)
[erdem@tux01] 5 1 96.3406  (best c=32.0, g=2.0, rate=96.3406)
[local] 7 -1 96.4493  (best c=128.0, g=0.5, rate=96.4493)
[erdem@tux07] 7 1 96.3768  (best c=128.0, g=0.5, rate=96.4493)
[erdem@tux06] 3 -9 94.6014  (best c=128.0, g=0.5, rate=96.4493)
```

Şekil 6.4 TuxNet konsol çıktısı.



Şekil 6.5 C ve ε parametrelerinin seçimi.

Sınama sonuçlarına göre en iyi başarım $C = 128$ ve $\varepsilon = 0.5$ değerleri için elde edilmiştir.

6.5.3 Verinin Ölçeklenmesi

Sınıflandırma için kullanılan öznitelikler farklı değer aralıklarında bulunabilir. Öznitelik değerlerinin kullanılmadan önce ölçeklenmesi başarımı önemli ölçüde arttırmaktadır (Hsu et al., 2003). Ölçeklemenin en önemli avantajı, büyük değerli özniteliklerin küçük değerli öznitelikleri gölgelemesini önlemesidir. Bir başka avantajı ise çekirdek fonksiyonlarının uygulanmasında yaşanabilecek nümerik problemlerin önüne geçilebilmesidir. Sınama, 22080 adet örnek için çapraz geçerlilik yöntemi ile yapılmıştır (k=10 alınmıştır).

Çizelge 6.9 Ölçeklemenin başarıma etkisi.

	Ham Veri	**Ölçeklenmiş Veri**
Doğruluk Oranı	70.14	**96.56**
Yanlış-Pozitif Oranı	0.62	**0.07**
F-Ölçütü	0.78	**0.97**
Eğitim Zamanı (CPU)	354.13	**12.54**
Sınama Zamanı (CPU)	112.49	**9.99**

Çizelge 6.9'teki sınama sonucunda görüldüğü üzere ölçeklenmiş verinin hem doğruluk oranı hem de çalışma zamanı ham veriye oranla daha iyidir.

6.6 Uygulamanın Başarımın Arttırılması

Sınıflandırma algoritmasından bağımsız olarak, ağ trafiği ve öznitelik çıkarma aşamalarında uygulama başarımını arttırmak için birkaç iyileştirme yapılmıştır.

Her paket için oturum bilgisi arandığı düşünülürse, bu işlemin olabilecek en kısa zamanda yapılması gerekliliği ortadadır. Oturum bilgisinin kıyım (hash) tablolarında tutulması sistemin çalışma hızını önemli ölçüde arttırmıştır.

Bazı öznitelik değerleri, belirli bir zaman (t) ve miktar (n) aralığındaki geçmiş oturum bilgilerinden yararlanılarak elde edilmektedir. Bu aralıklar uygulamaya parametre olarak verilmektedir. T ve n parametrelerinin uygun seçimi, bilgisayar ağının doğru örneklenmesi için önemlidir. Çizelge 6.10'de görüldüğü üzere farklı t ve n değerleri için sistem sınanmıştır. En iyi sonuç t=3 ve n=5000 için alınmıştır.

Çizelge 6.10 WindowTime ve WindowSize parametrelerinin başarıma etkisi.

		WindowSize (n)					
		100	*300*	*1000*	*2000*	*5000*	*10000*
WindowTime (t)	*1*	90.7406	91.9525	93.1091	93.41		
	2	90.4616	92.0926	93.2445	93.3705		
	3				93.5614	**93.75**	93.64
	4						
	5						93.72

Gerçek zamanlı ağ saldırı tespit sistemlerinde en büyük sorun, yüksek hızlı ağ trafiğinde tüm paketleri anlık olarak işlemenin zorluğudur. Bu sorunu aşmak için sistem olabildiğince yalın tasarlanmış olmalıdır. Sadece saldırı tespiti için anlamlı paketleri analiz etmek sistemin yüksek hızlardaki ağ trafiğini analiz etmesini sağlayacaktır. Saldırıların dış bilgisayar ağından iç bilgisayar ağına doğru geldiği göz önüne alındığında içeriden dışarıya doğru giden trafiğin analizi gereksiz olmaktadır. Ancak burada dikkat edilmesi gereken nokta tek bir paketin nereden nereye gittiği değil, oturumun ne yöne açıldığıdır. Oturum dış bilgisayar ağından iç bilgisayar ağına açıldı ise devam eden oturumda içeriden dışarıya gönderilen cevaplar da dikkate alınmalıdır. Çalışmada, sadece dış bilgisayar ağından açılan oturumlar incelendiğinde toplam incelenen trafiğin **%82,1** azaldığı görülmüştür. Toplam trafiğin **%17,9'u** ile yapılan analiz yaklaşık **13,4 kat** daha hızlı sonuç vermiş ve **%1,5** daha iyi başarım göstermiştir. KDD Cup veri kümesi ile sınanan saldırı tespit sistemlerinde yukarıda belirtilen sonuçlara ulaşmak mümkün olmamaktadır. Çünkü KDD Cup veri kümesinde oturumların hangi yöne açıldıklarına dair bir bilgi bulunmamaktadır.

6.7 Uygulamanın Diğer Sınıflandırma Algoritmaları ile Karşılaştırılması

Sistemin uygulanmasında sınıflandırma algoritması olarak destek vektör makinesi kullanılmıştır. Burada, makine öğrenmesi alanında kullanılan bazı önemli sınıflandırma algoritmalarına sistemden çıkan öznitelik değerleri verilerek tanıma başarıları karşılaştırılmıştır. Sınama için kullanılan diğer sınıflandırma algoritmaları;

- Random Forest
- NBTree

- J48 (C4.5)
- Naive Bayes
- K En Yakın Komşuluk (K Nearest Neighbor) (KNN)
- Çok Katmanlı Algılayıcı (Multilayer Perceptron) (MLP)

6.7.1 Random Forest

Birden fazla karar ağacından oluşan bir sınıflandırma algoritmasıdır. Leo Breiman ve Adele Culter tarafından geliştirilmiştir [25]. Random Forest algoritmasının avantajları aşağıda sıralanmıştır [24].

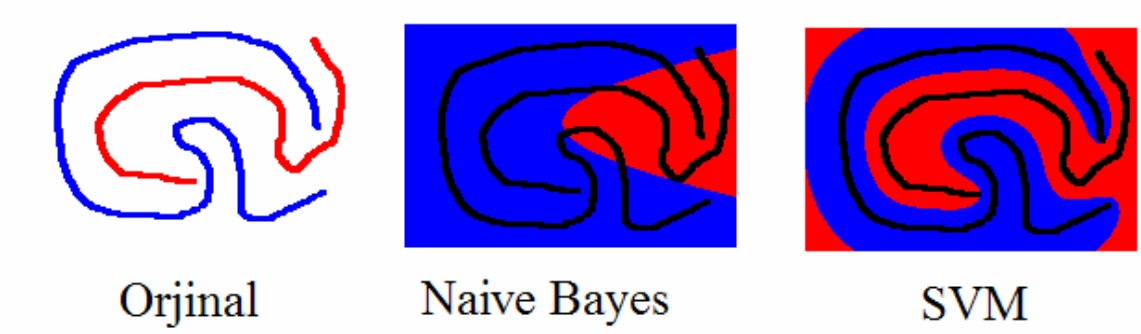
- Büyük sayıdaki verilerde yüksek başarımlı sonuçlar üretmektedir.
- Büyük sayıda giriş (öznitelik) değeri destekleyebilir.
- Öznitelikleri önem sırasına göre otomatik olarak algoritma içerisinde sıralar.
- Eğitim aşaması hızlıdır.

Random Forest, yüksek doğruluk oranlarını yakalarken, ayrıca tanıma için önemli özniteliklerin de seçimini kendi içinde yapmaktadır. Dong Seong Kim'in yaptığı çalışmada Random Forest, Destek Vektör Makinesi ve Yapay Sinir Ağlarına göre daha başarılı sonuçlar verdiği görülmektedir (Dong Seong Kim et al., 2006).

6.7.2 Naive Bayes

Bayes algoritmasının yalınlaştırılmış halidir. Değişkenler arasındaki bağıntı ile ilgilenmez. Bu durum birçok gerçek hayat uygulamasında başarılı sonuçlar vermektedir. Özellikle spam e-postaların tespitinde bu yaklaşım kullanılmaktadır.

Şekil 6.6'te iki sınıflı bir örneğin Naive Bayes ve SVM algoritmaları ile sınıflandırılma sonuçları verilmiştir. Görüldüğü gibi SVM doğrusal olmayan değerler için daha başarılı sonuçlar üretmektedir.



Şekil 6.6 Naive Bayes ile SVM algoritmalarının karşılaştırılması.

6.7.3 NBTree

Yapraklarda Naive Bayes algoritmasının uygulandığı bir çeşit karar ağacıdır. Naive Bayes algoritmasının başarımını arttırmak için kullanılan karma bir yapıdır. Yüksek miktardaki verilerin Naive Bayes yöntemi ile analizi için uygulanmaktadır (Kohavi, 1996).

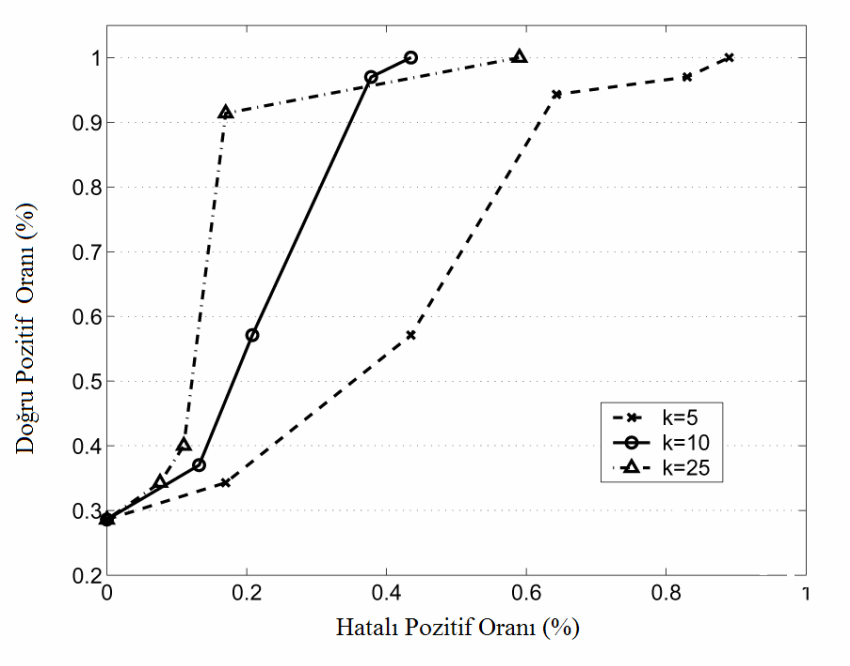
6.7.4 J48 (C4.5)

C4.5, ID3 algoritmasının geliştirilmiş bir türevidir. Quinlan tarafından geliştirilmiştir [26]. Weka sınıflandırma aracındaki gerçeklenmesi J48 olarak bilinir. C4.5 karar ağacının oluşturulması aşağıdaki gibi özetlenebilir [27].

1. Çıkış değerlerini en fazla farklılaştıran öznitelik seçilir.
2. Seçilen özniteliğin her değeri için farklı bir dal yaratılır.
3. Seçilen düğümdeki öznitelik değerlerini yansıtacak şekilde örnekler alt gruplara ayrılır.
4. Her alt grup için öznitelik seçimi durdurulur; Eğer
    a. Alt gruptaki tüm üyeler aynı çıkış değerini üretiyorsa, ağacın ilerlemesini durdurulur ve çıkış değeri olarak son belirlenen değeri atanır.
    b. Alt grupta tek düğüm kaldıysa veya ayırt edici öznitelikler belirlenemiyorsa ağacın ilerlemesi durdurulur.
5. 3. aşamada belirlenen her alt grup için yukarıdaki işlem tekrarlanır.

6.7.5 K En Yakın Komşuluk (KNN)

Genel olarak metin sınıflandırmada kullanılan KNN algoritması, Liao ve Vemuri tarafından saldırı tespiti amacıyla denenmiştir (Liao & Vemuri, 2002). Şekil 6.7'te KNN algoritmasının KDD Cup verisi ile sınanması sonucunda elde edilen alıcı işletim eğrisi (ROC) gösterilmektedir. Farklı K değerleri için başarım değişmektedir. Eğri altında kalan alan ne kadar büyükse doğruluk oranı da o kadar yüksektir. K=25 için algoritma en iyi değeri vermiştir.



Şekil 6.7 KNN algoritmasının saldırı tespiti başarısı (Liao & Vemuri, 2002).

6.7.6 Çok Katmanlı Algılayıcı (MLP)

MLP, bir tür yapay sinir ağıdır. Yapay sinir ağları saldırı tespitinde sıkça kullanılmaktadır. Yapay sinir ağları bu alanda başarılı sonuçlar vermektedir. Ancak, eğitim süresinin uzunluğu uygulamada sorunlar çıkarabilmektedir.

Mukkamala, SVM ve MLP algoritmalarının saldırı tespit başarılarını karşılaştırmıştır. Karşılaştırma sonucunda, eğitim zamanına göre SVM, MLP'ye göre daha başarılı sonuçlar

üretmektedir. Doğruluk oranları ise birbirlerine çok yakındır (%99 civarında). SVM az da olsa daha iyi sonuçlar üretmektedir. SVM, ikili sınıflandırma ile kısıtlı olsa da hızlı eğitilmesi, ölçeklenebilirliği ve genelleme kapasitesi artılarıdır (Mukkamala et al., 2002).

6.7.7 Karşılaştırma Sonuçları

Sınama için DARPA veri kümesinden seçilen 27601 adet örneğin %10'u (2760) eğitim, %90'ı (24841) sınama için kullanılmıştır. Sınama sonuçları Çizelge 6.11'de verilmiştir.

Çizelge 6.11 Diğer sınıflandırma algoritmaları ile karşılaştırılması.

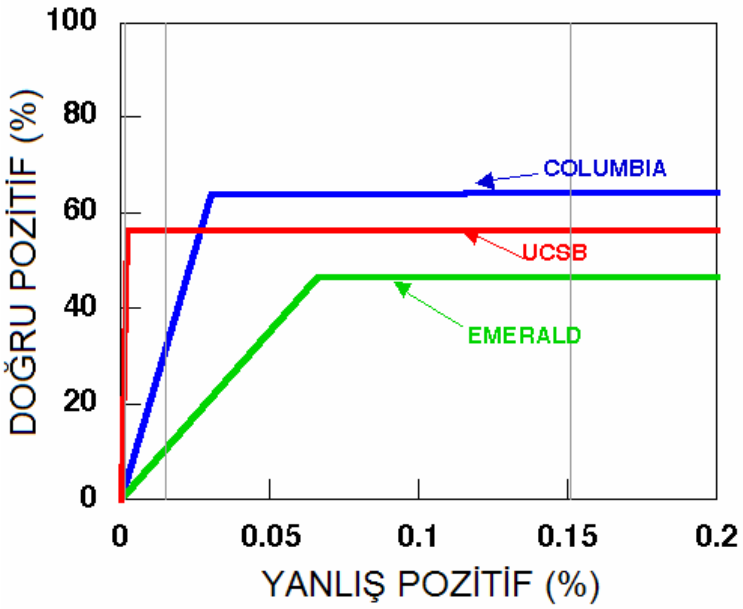
	Doğruluk Oranı (%)	**Hatalı Pozitif (%)**	**F-Ölçütü (%)**	**Eğitim İşlemci Zamanı (sn)**	**Sınama İşlemci Zamanı (sn)**
SVM	95.51	0.07	0.96	1.79	7.25
RandomForest	**96.86**	**0.05**	**0.97**	1.43	**0.22**
J48 (C4.5)	**96.69**	0.06	**0.97**	**0.36**	**0.10**
NBTree	96.42	0.07	**0.97**	16.28	0.68
Naive Bayes	95.70	0.09	0.96	**0.09**	1.63
KNN	96.33	**0.05**	**0.97**	**0.00**	74.03
MLP	96.34	**0.05**	**0.97**	234.77	3.74

Karşılaştırma sonuçlarına göre en iyi başarımı ağaç algoritmaları göstermiştir. RandomForest ve J48 algoritmalarında hem başarım hem de hız açısından diğer algoritmalara göre daha başarılı sonuçlara ulaşılmıştır. Karar ağaçlarının diğer algoritmalara göre daha basit yapıda olması, öznitelik ve parametre değerlendirme aşamalarını kendi içlerinde yapmaları en büyük artılarıdır. Naive Bayes iyi başarım göstermiş olmasına rağmen çok boyutlu veri için uygulamada öngörülemeyen sonuçlar üretebilmektedir. KNN algoritması en iyi eğitim zamanına sahip olmasına rağmen sınama aşaması gerçek zamanlı bir sistem için çok yavaştır. SOM, iyi eğitim ve sınama zamanlarına sahip olmasına rağmen başarım oranı diğer algoritmalara göre düşüktür. MLP algoritmasının uzun eğitim zamanı uygulamada büyük sorun oluşturmaktadır. Son olarak, SVM algoritması iyi bir doğruluk oranı yakalamış

olmasına rağmen SVM algoritmasının karmaşıklığı nedeniyle sınama zamanı pek de başarılı değildir.

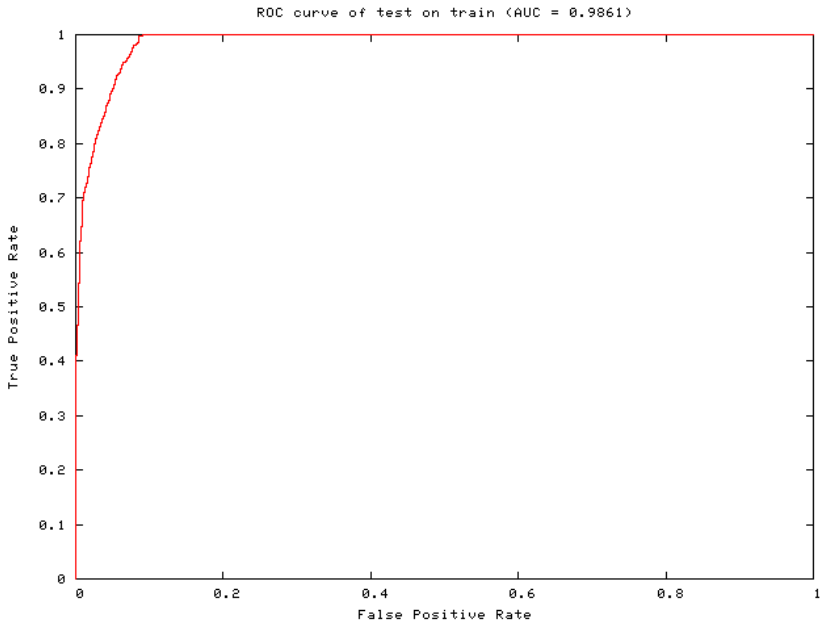
6.8 Uygulamanın Diğer Sistemler ile Karşılaştırılması

DARPA veri kümesindeki ham bilgisayar ağı verileri, DARPA veri kümesini hazırlayan çalışma grubu tarafından Columbia, UCSB ve EMERALD saldırı tespit sistemlerine verilmiştir. Elde ettikleri sonuçlar Şekil 6.8'deki alıcı işletim eğrisinde (ROC) karşılaştırılmıştır [31].



Şekil 6.8 DARPA verisi ile sınanan sistemlerin ROC eğrileri [31].

Tez çalışmasının başarımını karşılaştırmak için çalışmaya ait ROC eğrisi oluşturulmuştur. ROC eğrisinin çizilmesinde Tingfan Wu tarafından yazılmış "ROC Curve for Binary SVM" [32] eklentisinden yararlanılmıştır. Sistemin ROC eğrisi Şekil 6.9'de gösterilmiştir.



Şekil 6.9 Sistemin ROC eğrisi.

Sistemin başarısı ROC eğrisinin altında kalan alan ile ölçülür. Eğrinin altında kalan alan ne kadar büyükse doğruluk oranı o kadar iyidir denir. ROC altında kalan alan, toplam alana oranı 0.5 ile 1.0 arasında bir değer alır. Değer 0.975 ve daha üstü ise sonuç mükemmel sayılır. Şekil 6.8 ve Şekil 6.9'deki grafiklerin karşılaştırıldığında, tez çalışmasında uygulanan sistemin başarımının diğer saldırı tespit sistemlerine göre daha üst seviyede olduğu görülmektedir.

Son olarak, sisteme 3 haftada oluşturulan tüm DARPA veri kümesi verilmiştir. Veri çapraz geçerlilik sınamasından (k=10 alınmıştır.) geçirilmiştir. Sınıflandırma algoritması olarak C4.5 (J48) kullanılmıştır. Sonuç değerleri Çizelge 6.12'de gösterilmiştir.

Çizelge 6.12 Sistemin sınama sonuçları.

Parametre	**Değer**
Doğruluk Oranı (%)	91.33
Hata Oranı (%)	8.67
Eğitim Kümesi	96068
Sınama Kümesi	10674
Doğru Sınıflandırılan	9748
Yanlış Sınıflandırılan	925
Doğru Pozitif Oranı	0.96
Doğru Pozitif Sayısı	4138
Yanlış Pozitif Oranı	0.12
Yanlış Pozitif Sayısı	739
Doğru Negatif Oranı	0.88
Doğru Negatif Sayısı	5610
Yanlış Negatif Oranı	0.04
Yanlış Negatif Sayısı	186
Kesinlik	0.85
Geri Çağırım	0.96
F-Ölçütü	0.90
Geometrik Ortalama-1	0.90
Geometrik Ortalama-2	0.92
ROC alanı	0.95
Eğitim Zamanı (CPU)	156.70
Sınama Zamanı (CPU)	0.06

Sistem tüm DARPA verisi ile sınandığında doğruluk oranı **%91.33** olmuştur. ROC eğrisi altında kalan alan **0.95** olarak ölçülmüştür. Yanlış pozitif oranı **% 0.12** olmuştur. Elde edilen değerlere bakıldığında sistemin başarılı olduğu görülmektedir.

KDD Cup verisi ile yapılan sınamalarda daha yüksek tanıma başarıları elde edilmektedir. Bunun nedeni eğitilen verinin tamamıyla saldırıları kapsıyor olmasıdır. Oysaki DARPA veri kümesinde saldırı yapılan bir günde ortalama 8 saldırı gerçekleştirilmiştir. Geri kalan trafik ise saldırı içermeyen normal trafiktir. Bu senaryo daha gerçekçidir ve uygulamaya yönelik daha doğru sonuçlar vermektedir.

Bir başka farklılık da KDD Cup veri kümesinin gerçek zamanlı değil, toplu bir şekilde işlenmiş olmasıdır.

DARPA veri kümesi ağ trafiğinden gelen paketler ve sunuculardan gelen günlükler olmak üzere ikiye ayrılmıştır. KDD Cup veri kümesi, bu iki farklı veriyi aynı anda birleştirerek kullanmıştır. Sınanan sistemlerde genelde yapılan yanlışlık, tasarlanan sistemin sunucu veya ağ tabanlı olmasına rağmen KDD Cup verisinin tamamının kullanılmasıdır. Ancak, KDD Cup verisi her iki yaklaşıma da ait verileri içermektedir ve doğru bir sınama için bu iki verinin ayrılması gerekmektedir.

7. SONUÇ

Tez çalışmasında gerçek zamanlı veriler yardımı ile karar veren bir bilgisayar ağı saldırı tespit sistemi tasarlanmış ve gerçeklenmiştir. Anormallik ve imza tabanlı yaklaşımlar birleştirilerek karma bir yapı oluşturulmuştur. İmza tabanlı yaklaşım için Snort yazılımından yararlanılmıştır. Tez çalışması, Snort yazılımına ön işlemci eklentisi olacak şekilde tasarlanmıştır. Anormallik tespitinde bilgisayar ağı trafiği "normal" ve "kötü" olmak üzere iki sınıfta incelenmiştir. Anormallik tespiti için bilgisayar ağı trafiğinden belli başlı öznitelik değerleri çıkarılmıştır. Sınıflandırma için çeşitli algoritmalar denenmiş ve başarımları karşılaştırılmıştır. Tez çalışmasında elde edilen sonuçlar aşağıda sıralanmıştır.

1. İmza tabanlı saldırı tespiti yaklaşımının gerçeklenmesinde Snort yazılımından yararlanılmıştır. Böylece sisteme,

    a. Birden çok ağ protokolünü destekleme
    b. Taşınabilirlik
    c. Standart bir yapı
    d. Güncel imza veritabanı
    e. Zengin uyarı çıkış eklentileri

   Özellikleri kazandırılmıştır.

2. Öznitelikler önem derecesine göre sıralanmıştır. En önemli 4 adet öznitelik kullanılarak hemen hemen aynı başarım 23 öznitelikle yapılan sınamanın **%22,25**'i zamanda elde edilmiştir
3. Mukkamala'nın (Mukkamala et al., 2002) çıkardığı özniteliklerden farklı olarak çıkarılan yeni öznitelikler daha başarılı sonuçlar üretmiştir.
4. Oturum bilgileri program içerisinde kıyım (hash) tablolarında saklanarak arama süresi en aza indirilmiştir.
5. Uygulamada kullanılan destek vektör makinesinde yapılan iyileştirmeler sistemin hem hızını hem de doğruluk oranını arttırmıştır.
6. Sistemin ürettiği öznitelik değerleri normalize edilerek doğruluk oranı **%36,24** arttırılmıştır. Ham veri ile yapılan eğitim zamanı **%96,46**, sınama zamanı **%91,12** azaltılmıştır.

7. Sistemin bilgisayar ağ trafiğini daha geniş bir bakış açısı ile incelemesi için geçmiş bağlantı bilgilerini tutan ara bellek yapıları oluşturulmuştur. Bu yapıların kapsamları iyileştirilerek yaklaşık **%3,6** daha iyi tanıma sağlanmıştır.
8. Sadece, saldırı tespiti için incelenen bilgisayar ağına doğru açılan bağlantılar incelenerek bilgisayar ağında oluşturulan toplam trafiğin **%82,1**'i göz ardı edilmiştir. Toplam trafiğin sadece **%17,9**'u incelenerek **%1,5** daha iyi tanıma sağlanmıştır. Sistemin çalışma hızı ise **13,4 kat** artmıştır. Bu iyileştirme sadece KDD Cup verisi ile sınanan sistemlerde ulaşılamayacak bir sonuçtur. Çünkü KDD Cup verisinde bağlantıların yönüne ait bir bilgi bulunmamaktadır.
9. Tez çalışması birçok sınıflandırma algoritması ile sınanmıştır. En iyi sonucu J48 (C4.5) ve Random Forest karar ağacı algoritmaları vermiştir. Uygulamada bu algoritmaların kullanımı sistemin çalışma hızını çok önemli ölçüde arttırmaktadır.
10. Uygulama, diğer saldırı tespit sistemleri ile de karşılaştırılmıştır. Diğer sistemlere göre doğruluk oranının daha iyi olduğu görülmüştür.

Yapılan sınama ve analizler sonucunda sistemde ileriye yönelik aşağıdaki değişikliklerin yapılması öngörülmektedir.

1. UDP protokolünün durum bilgili olarak desteklenmesi.
2. Her protokolün (TCP, UDP, ICMP) ayrı sınıflandırılması.
3. J48 ve Random Forest algoritmaların uygulamaya bütünleştirilmesi.
4. Snort'ta bulunan saldırı önleme eklentisi etkinleştirilerek sisteme saldırı önleme kabiliyetinin kazandırılması.

KAYNAKLAR

Anderson,D., Frivold,T., and Valdes,A. (1995). *Next-generation Intrusion Detection Expert System (NIDES) A Summary*.

Anderson,J.P. (1980). Computer security threat monitoring and surveillance (Technical Report). *Washington, PA*.

Arboleda,A.F. and Bedón,B.E. (2006). Snort Diagrams for Developers.

Axelsson,S. (2000). Intrusion detection systems: A survey and taxonomy. *Chalmers University, March*.

Blomqvist,D. and Skantze,J. (1995). *Intrusion Detection: A Study*. Technical Report DoCS 95/62, Department of Computer Systems, Uppsala University.

Bülent Sankur (2005). Biliþim Sözlüðü 2005. Pusula Yayýncýlýk.

Debar,H., Becker,M., & Siboni,D. (1992). A neural network component for an intrusion detection system. *Research in Security and Privacy, 1992 Proceedings , 1992 IEEE Computer Society Symposium on,*240-250.

Denning,D.E. (1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering, 13*,222-232.

Depren,O., Topallar,M., Anarim,E., & Ciliz,M.K. (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems With Applications, 29*,713-722.

Dong Seong Kim, Sang Min Lee, & Jong Sou Park (2006). Building Lightweight Intrusion Detection System Based on Random Forest. *Lecture Notes in Computer Science, 3973*,224-230.

Hochberg,J., Jackson,K., Stallings,C., McClary,J.F., DuBois,D., & Ford,J. (1993). NADIR: an automated system for detecting network intrusion and misuse. *Computers and Security, 12*,235-248.

Hsu,C.W., Chang,C.C., & Lin,C.J. (2003). A practical guide to support vector classification. *National Taiwan University, Tech Rep , July*.

Ilgun,K. (1992). USTAT: A Real-Time Intrusion Detection System for UNIX. *Master's thesis, Computer Science Department, University of California, Santa Barbara, July*.

Kepenekci,B. & Akar,G.B. (2004). Face classification with support vector machine. *Signal Processing and Communications Applications Conference, 2004 Proceedings of the IEEE 12th,*583-586.

Kohavi,R. (1996). Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining,*202-207.

Kohavi,R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, 14*,1137-1143.

Lee,W., Stolfo,S.J., Chan,P.K., Eskin,E., Fan,W., Miller,M., Hershkop,S., & Zhang,J. (2001). Real time data mining-based intrusion detection. *DARPA Information Survivability Conference & Exposition II, 2001 DISCEX'01 Proceedings, 1*.

Liao,Y. & Vemuri,V.R. (2002). Use of K-Nearest Neighbor classifier for intrusion detection. *Computers and Security, 21*,439-448.

Lunt,T.F. (1989). Real-time intrusion detection., p. 348-353.

McHugh,J. (2001). Intrusion and intrusion detection. *International Journal of Information Security, 1*,14-35.

Mukkamala,S., Janoski,G., & Sung,A. (2002). Intrusion Detection using Neural Networks and Support Vector Machines. *Neural Networks, 2002 IJCNN'02 Proceedings of the 2002 International Joint Conference on, 2*.

Musicant,D., Kumar,V., & Ozgur,A. (2003). Optimizing F-measure with support vector machines. *Proceedings of the International Florida Artificial Intelligence Research Society Conference,*356-360.

P.A.Porras & P.G.Neumann (1997). EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. *Proc 20th {NIST}-{NCSC} National Information Systems Security Conference,*353-365.

Paxson,V. (1988). Bro: A System for Detecting Network Intruders in Real-Time. *Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, USA*.

Ptacek,T.H. and Newsham,T.N. (1998). Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc., January 1998.

Rehman,R. (2003). *Intrusion Detection With Snort: Advanced Ids Techniques Using Snort, Apache, MySQL, PHP, and Acid*. Prentice Hall PTR.

Schneier,B. (1993). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc. New York, NY, USA.

Schultz,G., Endorf,C., and Mellander,J. (2003). *Intrusion Detection*. McGraw-Hill Osborne Media.

Sebring,M., Shellhouse,E., Hanna,M., & Whitehurst,R. (1988). Expert Systems in Intrusion Detection: A Case Study. *Proceedings of the 11th National Computer Security Conference, 81*.

Smaha,S.E., Inc,T.A.S., & Austin,T.X. (1988). Haystack: an intrusion detection system. *Aerospace Computer Security Applications Conference, 1988 , Fourth,*37-44.

Stallings,W. (2006). *Network Security Essentials: Applications and Standards*. Prentice Hall.

Vigna,G. (1999). NetSTAT: A network-based intrusion detection system. *Journal of Computer Security, 7*,37-71.

Yahsieli R, Ciliz K, and Anarim E (4 A.D.). Saldiri Tespit Sistemi SNORT ile Yakalanan Saldirilarin SOM Yapisi Icinde Kümelestirilmesi. *Savunma Teknolojileri Kongresi Bildiriler Kitabý (SAVTEK 2004)*. Ankara: p. 261-268.

INTERNET KAYNAKLARI

[1] http://www.tdk.gov.tr
[2] http://www.e-hack.org/index.php?id=56
[3] http://www.snort.org (17.02.2007)
[4] http://www.faqs.org/rfcs/rfc1918.html
[5] http://www.faqs.org/rfcs/rfc862.html
[6] http://www.faqs.org/rfcs/rfc864.html
[7] http://www.tech-faq.com/packet-fragmentation.shtml
[8] http://www.faqs.org/rfcs/rfc959.html
[9] http://httpd.apache.org/
[10] http://www.iis.net
[11] http://news.netcraft.com/archives/web_server_survey.html
[12] http://www.snort.org/about_snort/ (23.01.2007)
[13] http://kdd.ics.uci.edu//databases/kddcup99/task.html (28.10.1999)
[14] http://www.ll.mit.edu/IST/ideval/data/data_index.html (2006)
[15] http://www.csie.ntu.edu.tw/~cjlin/libsvm/ (17.11.2006)
[16] http://svmlight.joachims.org/ (02.09.2004)
[17] http://en.opensuse.org/ (07.12.2006)
[18] http://gcc.gnu.org/ (07.12.2006)
[19] http://www.iana.org/assignments/port-numbers (01.03.2007)
[20] http://www.rl.af.mil/ (01.03.2007)
[21] http://www.ll.mit.edu/IST/ideval/docs/1999/Network_Topology.gif (01.02.1999)
[22] http://www.ll.mit.edu/IST/ideval/docs/1999/attackDB.html (01.02.1999)
[23] http://www.svms.org/parameters/ (01.02.2007)
[24] http://www.stat.berkeley.edu/~breiman/RandomForests/ (01.02.2007)
[25] http://en.wikipedia.org/wiki/Random_forest (01.02.2007)
[26] http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5/tutorial.html (01.02.2007)
[27] http://grb.mnsu.edu/grbts/doc/manual/J48_Decision_Trees.html (01.02.2007)
[28] http://www.cs.waikato.ac.nz/~ml/weka/ (01.02.2007)
[29] http://wekaclassalgos.sourceforge.net/ (04.02.2007)
[30] http://www.cs.iastate.edu/~yasser/wlsvm/ (04.02.2007)
[31] http://www1.cs.columbia.edu/~sal/JAM/PROJECT/MIT/sld014.htm (06.02.2007)
[32] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/#2 (06.02.2007)
[33] http://en.wikipedia.org/wiki/EXEC_8
[34] http://en.wikipedia.org/wiki/Multics

EKLER

Ek 1 Snort Ön İşlemcisi Oluşturmak İçin İzlenmesi Gereken Adımlar

Aşağıda anlatılan kurulum i686 mimarisindeki bir bilgisayarda; OpenSuse Linux 10.2 işletim sistemi, gcc 4.1.0 derleyici ve snort-2.6.1.2 yazılımı ile gerçeklenmiştir.

1. http://www.snort.org/dl/current/ adresinden en son Snort yazılımını indiriniz.

2. Uygun bir dizine açınız (Örn: /root/snortinstall/).
   ```
   root:~ # tar -xzvf /root/snortinstall/snort-2.6.1.2.tar.gz
   ```

3. `templates/` klasöründeki `spp_template.c` ve `spp_template.h` şablon dosyalarını kopyalarak `src/preprocessors/` dizinin içine yapıştırınız. Dilerseniz ön işlemcinin isimini `template` dışında başka bir isim olarak belirleyebilirsiniz. Ancak `spp_` ön ekini kesinlikle değiştirmeyiniz. Eğer isim değişikliği yaptıysanız bu aşamadan sonra gördüğünüz tüm `template` ifadelerini kendinizin belirlediği isim ile değiştirmelisiniz.

4. `spp_template.h` dosyasını bir kelime işlemcisi ile (örn: vi, gedit) açınız. Dosyanın içeriği basitçe aşağıdaki gibi olmalıdır. Dosyanızdaki açıklama satırlarını olduğu gibi bırakabilirsiniz Eğer isim değişikliği yapıldıysa her `template` ifadesi yerine kendi belirlediğimiz ismi yazınız.

   ```
   #ifndef __SPP_TEMPLATE_H__
   #define __SPP_TEMPLATE_H__
   void SetupTemplate();
   #endif
   ```

5. `spp_template.c` dosyasını açınız ve açıklama satırının hemen altına aşağıdaki kod parçasını ekleyiniz.

   ```
   #ifdef HAVE_CONFIG_H
   #include "config.h"
   #endif
   #include "decode.h"
   ```

6. `RegisterPreprocessor()` fonksiyonunun ilk parametresi ön işlemcinin adıdır. Bu değer daha sonra `snort.conf` dosyasına eklenecektir.

```
void SetupTemplate()
{ ..
RegisterPreprocessor("keyword", TemplateInit);
..}
```

7. `TemplateInit()` fonksiyonundaki aşağıda belirtilen satırları uygun şekilde değiştiriniz.

```
static void TemplateInit(u_char *args)
{
      DebugMessage(DEBUG_PLUGIN,"On Islemcı: Template
      Etkinlestirildi\n");
      ParseTemplateArgs(args);
      AddFuncToPreprocList(PreprocFunction, PRIORITY_FIRST, 1);
      AddFuncToPreprocCleanExitList(TemplateCleanExit,NULL,
      PRIORITY_APPLICATION, 0);
      AddFuncToPreprocRestartList(TemplateRestart,NULL,
      PRIORITY_APPLICATION, 0);

}
```

8. `PreprocFunction()` ana fonksiyonunuzdur ve her paket için çalıştırılır. Buraya örnek bir kod eklenebilir.

```
static void PreprocFunction(Packet *p, void *context){
printf("Merhaba! On Islemci cagrildi..\n");
}
```

9. `src/plugbase.c` dosyasını açın ve diğer ön işlemci tanımlarının altına aşağıdaki satırı ekleyin.

```
#include "preprocessors/spp_template.h"
```

10. `InitPreprocessors()` fonksiyonunu bulun ve diğer ön işlemci fonksiyonlarının altına aşağıdaki satırı ekleyiniz.

```
SetupTemplate();
```

11. `src/preprocessors/Makefile.in` dosyasını açınız. `libspp_a_SOURCES` değerinin sonuna `spp_template.c spp_template.h` ifadesini ekleyiniz. am_libspp_a_`OBJECTS` değerinin sonuna da `spp_template.$(OBJEXT)` ifadesini ekleyiniz. Böylece ön işlemci fonksiyonumuz Snort ile birlikte derlenecektir.

12. Aşağıdaki komutları sırasıyla kullanarak kodu; yapılandırınız, derleyiniz ve yükleyiniz.

```
./configure
make
make install
```

13. Snort'a ek özellikler kazandırmak istiyorsanız (örn. veritabanı desteği) `configure` komutuna parametre eklemelisiniz. ./`configure` --help komutu ile gerekli parametreleri öğrenebilirsiniz.

14. Kodunuz başarılı bir şekilde derlendikten ve yüklendikten sonra `/etc/snort/snort.conf` dosyasını açınız ve "Configure preprocessors" kısmına aşağıdaki kodu ekleyiniz. Buradaki `keyword` daha önce spp_`template`.c kaynak kodundaki `RegisterPreprocessor()` fonksiyonuna verdiğimiz birinci parametre olduğunu unutmayınız.

```
preprocessor keyword
```

`Snort.conf` doyasının düzenlenmesi bu değişiklikle sınırlı değildir. Eklediğiniz özelliklere göre yapılandırma dosyası değişiklik gösterebilir.

15. Son olarak aşağıdaki komut ile Snort'u çalıştırınız. Gelen her paket için `PreprocFunction()` fonksiyonu çağrılacaktır.

```
root:~ # /usr/local/bin/snort –c /etc/snort/snort.conf –l \
/var/log/snort/
```

Eğer bazı ön işlemciler için hata mesajı verirse `snort.conf` dosyasında tanımlandıkları yerleri bulup açıklama satırı ile kapatınız. Yukarıdaki komutta belirtilmeyen parametreler için varsayılan değerler kullanılmaktadır. Eğer farklı bir

yapılandırmanız varsa bunu belirtmelisiniz. Örneğin, analiz etmek istediğiniz ağ arayüzü `eth0` değil de `eth1` ise `-i eth1` parametresini komuta eklemelisiniz.

ÖZGEÇMİŞ

Doğum tarihi	10.12.1981	
Doğum yeri	Cide	
Lise	1996–1999	Bartın Davutfırıncıoğlu Anadolu Lisesi
Lisans	1999–2003	Yıldız Teknik Üniversitesi Elektrik-Elektronik Fakültesi Bilgisayar Mühendisliği Bölümü

Çalıştığı Kurumlar

	2002–2003	Borusan Elektronik (Otomax.com) Web Uygulama Geliştiricisi
	2003-Devam ediyor	Yıldız Teknik Üniversitesi Bilgi İşlem Daire Başkanlığı Araştırma Görevlisi