İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GEZGİN ETMENLER VE DOĞADAN ESİNLENEN SEZGİSELLER KULLANARAK DAĞITIK BİLGİSAYAR GÜVENLİĞİNİN SAĞLANMASI

DOKTORA TEZİ

Uğur AKYAZI

Anabilim Dalı : Bilgisayar Mühendisliği

Programı : Bilgisayar Mühendisliği

NİSAN 2011

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GEZGİN ETMENLER VE DOĞADAN ESİNLENEN SEZGİSELLER KULLANARAK DAĞITIK BİLGİSAYAR GÜVENLİĞİNİN SAĞLANMASI

DOKTORA TEZİ
Uğur AKYAZI
(504032503)

Tezin Enstitüye Verildiği Tarih : 15 Eylül 2010
Tezin Savunulduğu Tarih : 20 Nisan 2011

Tez Danışmanı : Yrd. Doç. Dr. A. Şima Etaner UYAR (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Nadia ERDOĞAN (İTÜ)
Prof. Dr. Bülent ÖRENCİK (TÜBİTAK)
Yrd. Doç. Dr. Güray YILMAZ (HHO)
Yrd. Doç. Dr. Sanem Sarıel TALAY (İTÜ)

NİSAN 2011

Eşime ve çocuklarıma,

ÖNSÖZ

Tez çalışmam süresince hiç bir koşulda desteklerini benden esirgemeyen ve yol gösteren danışmanım Yrd. Doç. Dr. A. Şima Etaner Uyar'a, iş hayatımdaki desteklerinden dolayı Hava Harp Okulu Bilgisayar Mühendisliği Bölüm Başkanı Yrd. Doç. Dr. Hv. Müh. Yb. Güray Yılmaz'a, büyük bir sabırla ve fedakârlıkla tez çalışmamı başarıyla bitirmemi destekleyen eşime, kızıma ve tezimin sonunu bana müjdeleyen oğluma, manevi desteklerini her zaman arkamda hissettiğim annem, babam ve kardeşime en içten teşekkürlerimi sunarım.

Eylül 2010 Uğur Akyazı

İÇİNDEKİLER

Sayfa

KISALTMALAR

Ab : Antikor
Ag : Antijen
AIS : Yapay Bağışıklık Sistemi
ANN : Yapay Sinir Ağları
BIS : Biyolojik Bağışıklık Sistemi
BSM : Temel Güvenlik Modu
DARPA : Savunma İleri Araştırma Projeleri Ajansı
DDoS : Dağıtık Hizmetin Engellenmesi Saldırısı
DNS : Alan Adı Sunucusu
DoS : Hizmetin Engellenmesi Saldırısı
dSBS : Dağıtık Sızma Belirleme Sistemi
EA : Evrimsel Algoritma
FN : Yanlış Negatif
FP : Yanlış Pozitif
GP : Genetik Programlama
HHM : Gizlenmiş Markov Modelleri
ICMP : İnternet Kontrol Mesajı Protokolü
IP : İnternet Protokolü
Jade : Java Etmen Geliştirme Çerçevesi
KDD CUP : Veritabanında Bilgi Keşfi Kupası
kNN : K En Yakın Komşu Algoritması
MA : Gezgin Etmen
MAD : Gezgin Etmen Göndericisi
MİB : Merkezi İşlemci Birimi
MOEA : Çok-amaçlı Evrimsel Algoritma
PF : Pareto Front
R2L : Uzaktan Yerele Saldırılar
SA : Durağan Etmen
SBS : Sızma Belirleme Sistemi
SOM : Kendisi Organize olan Haritalar
SVM : Destek Vektörü Makinesi
TCP : Transmisyon Kontrol Protokolü
TN : Doğru Negatif
TP : Doğru Pozitif
U2R : Kullanıcıdan Yöneticiye Saldırılar
UDP : Kullanıcı Datagram Protokolü
VHL : Ziyaret Edilen Konakçı Listesi

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

GEZGİN ETMENLER VE DOĞADAN ESİNLENEN SEZGİSELLER KULLANARAK DAĞITIK BİLGİSAYAR GÜVENLİĞİNİN SAĞLANMASI

ÖZET

Sızma, bir kaynağın bilgi güvenliği temelini oluşturan gizlilik, bütünlük ve kullanılabilirliğini tehlikeye sokmaya çalışan bir olay olarak tanımlanmaktadır. Bir organizasyonu tam koruma altına almak için, ağı düzenli olarak sızma girişimleri için gözlemlemek gerekmektedir. Sızma Belirleme Sistemi'nin (SBS) görevi hem sisteme erişmeye çalışan yetkisiz bir sızıcıyı, hem de sistem kaynaklarını kötüye kullanan yetkili bir kullanıcıyı belirlemektir. SBS'ler, beklenen bilgisayar davranışlardan olan sapmaları veya belirli ağ trafiği paternlerini aramak amacıyla yerel izleme verilerini işleyen veya ağ trafiğini inceleyen güvenlik araçlarıdır.

Genelde, Hizmetin Engellenmesi Saldırısında (DoS) olduğu gibi, bir saldırıcı öncelikle yazılım açıklarını suiistimal ederek tek bir kullanıcıya erişim sağlar, daha sonra önceden ele geçirdiği konakçı vasıtasıyla ağdaki diğer konakçılara girmeyi dener. DoS saldırısının amacı hedef sistemin normalde sunduğu hizmetleri sunamaz hale getirmektir. Dağıtık Hizmetin Reddedilmesi Saldırısında (DDoS), aynı anda çok sayıda kaynak tarafından tek bir hedefe saldırı düzenlenir. Bu saldırılar, genelde önceden başkasının kendilerini bir saldırı başlatmak için kullanabilmesi amacıyla ele geçirilmiş olan bilgisayarları kullanırlar. Bu zombi bilgisayarlar saldırının orta katmanında rol alırlar.

Bu tez çalışmasında, gezgin etmenler ve doğadan esinlenen algoritmalar kullanarak DDoS-öncesi dağıtık saldırıları orta katmanda dağıtık olarak belirlemek ve saldırı başarıya ulaşmadan önce güvenlik yöneticilerini haberdar etmek amaçlanmıştır. Bu kapsamda gezgin etmenler yardımıyla dağıtık yapının kurulması ve doğadan esinlenen algoritmalar yardımıyla anomali-temelli bir SBS oluşturulması ayrı ayrı gerçeklenmiş ve en sonunda, bu iki sistem birleştirilerek yeni saldırıları düşük yanlış pozitif oranlarıyla belirleyebilen, adaptif olarak SBS'i güncelleme imkanına sahip, merkezi üniteden bağımsız çalışabildiği için yüksek güvenirlik sahibi, sızma belirleme işlemini gezgin etmenler vasıtasıyla yaptığı için ağ yükünü hafifleten ve özellikle DDoS saldırılarını kötüye kullanıldıklarından habersiz olan ve "botnet" olarak nitelendirilen orta katmanda dağıtık olarak başarıyla belirleyebilen bir SBS geliştirilmiştir.

İlk aşamada, MIT DARPA 2000 LLS_DDOS 1.0 verisetindeki DDoS saldırılarını tespit edebilmek için altı farklı dağıtık sızma belirleme yöntemi tasarlandı ve simule-gerçek zamanlı test ortamında saldırıların gerçekleşme zaman bilgilerini dikkate alarak test edildi. Bu imza-temelli SBS yöntemlerinin birincisi dışında hepsinde gezgin etmenler kullanıldı. Yöntem_1'de Mobile Agent'lar kullanılmaksızın merkezi bir dağıtık SBS yer almakta olup bir saat içerisinde farklı konakçılardan aynı kaynaklı, aynı tipte bir sızma-şüphe mesajı alınırsa dağıtık bir sızma yapıldığı kararına varılmakta ve güvenlik yöneticisi haberdar edilmektedir. Yöntem_2'de dağıtık sızma olup olmadığı kararını, yukarıda belirtilen şartlar oluştuğunda

MainAgent tarafından yaratılan ve sadece listesindeki iki adet konakçıya giderek ilgili verileri yerinde inceleyen Mobile Agent'lar vermektedirler. Yöntem_3'de, MainAgent, Mobile Agent yaratmak için gerekli şartları sağlayan mesaj sayısının iki olmasını beklemez ve ilgili etmenleri sisteme yollar. Yöntem_4'de ise, Mobile Agent'lar merkez tarafından değil de konakçılardaki Static Agent'lar tarafından merkezle koordineli olarak yaratılırlar. Yöntem_5'te, Mobile Agent'lar sızma-şüphelerini MainAgent ve diğer konakçılarla koordine kurmaksızın belirleyen Static Agent'lar tarafından yaratılmakta ve yollanmaktadır. MainAgent hiç kullanılmadığından bu yöntem tam-dağıtık bir yöntemdir. Mobile Agent dağıtık sızma kararını verebilmek için bütün ağı rastgele bir sırada dolaşmaktadır. Yöntem_6'da, Yöntem_2'nin kısa ortalama belirleme süresi avantajı ile Yöntem_5'in tam-dağıtık yapısı birleştirilerek yüksek güvenirlik ve kısa belirleme süresi sahibi bir sistem geliştirilmiştir. Bu sistem, normalde mod-1'de (Yöntem_2) çalışırken, merkezi ünite kullanılamaz olduğunda mod-2'ye (Yöntem_5) geçmektedir.

Veri setinde yer alan DDoS saldırısının ilk üç aşaması doğru ve hızlı olarak belirlenerek, son iki aşaması gerçekleşmeden gerekli tedbirlerin alınması için güvenlik yöneticisinin uyarılması hedeflenmiştir. Yapılan yirmi adet koşturma sonucunda her bir yöntemin anılan sızma aşamalarını ortalama belirleme süreleri, ağdaki yük ve güvenirlik özellikleri ayrı ayrı değerlendirilmiştir. Yöntem_4, merkezi üniteden tam-bağımsız olması ve düşük ağ yüküne sahip olması nedeniyle tezin bu bölümü için en iyi yöntem olarak değerlendirilmiştir.

İkinci aşamada, SBS yapısı ve biyolojik bağışıklık sistemi arasındaki benzerlikten dolayı anomali-temelli sızma belirleme yöntemi olarak yapay bağışıklık sistemi kullanılmıştır. MIT DARPA 2000 LLS_DDOS 1.0 verisetindeki DDoS saldırılarını belirlerken daha iyi doğru ve yanlış pozitif oranları elde etmek amacıyla çok-amaçlı evrimsel algoritmadan esinlenen bir yapay bağışıklık sistemi olan jREMISA çalışmasına yeni geliştirmeler eklendi. Bu geliştirmeler: r-sürekli değerlendirme yönteminin eklenmesi, Negatif Seleksiyon ve Klonlama Seleksiyon'da değişiklikler yapılması, genel konsept korunurken ikinci hedefin yeniden tanımlanması olarak özetlenebilir.

Geliştirilmiş-jREMISA üzerinde daha iyi doğru ve yanlış pozitif sonuçlar verecek en iyi parametre grubunu bulabilmek için benzerlik eşiği değerleri, r-sürekli değerleri ve birincil popülasyon büyüklüklerinden oluşan parametrelerin değişik ayarlamaları ile üç farklı test yapıldı. En sonunda, orjinal ve geliştirilmiş-jREMISA, önceden belirlenen en iyi parametre grubu kullanılarak karşılaştırıldı. Geliştirilen algoritmaya ait olan %100 doğru pozitif oranı ve %0 yanlış pozitif oranı, bir anomali sızma belirleme sistemi olarak kaydadeğer bir başarıdır.

Daha sonra, geliştirilmiş-jREMISA'nın performansını diğer benzer çalışmalarla karşılaştırabilmek için literatürde yaygın olarak kullanılan 1999 DARPA SBS veriseti günlerinin farklı bileşimleri kullanılarak deneyler yapıldı. Deneylerde yaklaşık %100 doğru pozitif oranı ve yaklaşık %0 yanlış pozitif oranı başarısı elde edilmiştir. Aynı veriseti üzerinde yapılan diğer çalışmalarla karşılaştırıldığında, geliştirilmiş-jREMISA'nın hepsinden daha iyi TP ve FP değerleri olduğunu gözlemlenmiştir.

Son olarak, ayrı ayrı geliştirilen bu iki yapı birleştirilerek tez çalışmasında hedeflenen sistem gerçeklenmiştir. Dağıtık sistemde her bir konakçıda yer alan imza-temelli bir SBS olan Snort tarafından oluşturulan sızma-şüphesi alarmlarının yerini

anomali-temelli SBS olan geliştirilmiş-jREMISA'nın oluşturduğu alarmlar aldı. İmza-temelli sistemde geliştirilen Yöntem_2, Yöntem_3 ve Yöntem_4'ün anomali-temelli sistemde diğer yöntemlerden çok farklı işlevleri olmadığına karar verilerek vazgeçilmiş, diğer yöntemler anomali-temelli belirleme yapısına göre yeniden ayarlanmıştır. Sisteme güvenlik yöneticisi tarafından sonradan eklenen saldırı tiplerinin tanınabilmesi için SBS'nin sürekli adaptif olarak çalışması sağlanmıştır. Mobile Agent'lar diğer konakçıların alarm dosyalarını sorgularken benzer bir anomali-tipi sızma-şüphesi paketini bulduğu konakçının DDoS seviyesini artırmakta ve her seferinde konakçılara bir zararlı yazılımın yüklenip yüklenmediğini ayrıca kontrol etmektedir. Güvenlik yöneticisi, bütün bu saldırı aşamalarından güvenli bir alanda kurulduğu varsayılan AlarmAgent aracılığıyla haberdar edilmektedir.

MIT DARPA 2000 LLS_DDOS verisetinin 1.0 versiyonu ve daha akıllı bir saldırının senaryo edildiği 2.0.2 versiyonu kullanılarak yapılan testler başarıyla sonuçlanmıştır. DDoS saldırıları gerçekleşmeden önce güvenlik yöneticisini uyaran, gerektiğinde merkezi üniteden tam bağımsız çalışabilme özelliğine sahip, ağda gereğinden fazla yük oluşturmayan, adaptif bir dağıtık SBS gerçeklenmiştir. "DDoS saldırılarının dağıtık olarak belirlenmesi", "Gezgin etmenler kullanarak sızmaların dağıtık olarak belirlenmesi" ve "Anomali-temelli SBS" konularında çalışmalar olmasına rağmen, DDoS saldırıları öncesi dağıtık hazırlık hareketlerinin zombi bilgisayarlardan oluşan orta katmanda yüksek doğruluk oranlarıyla çalışan bir anomali-temelli SBS yapısıyla belirlenmesi ilk defa bu çalışmada gerçekleştirilmiştir.

DISTRIBUTED COMPUTER SECURITY USING MOBILE AGENTS AND NATURE INSPIRED ALGORITHMS

SUMMARY

An intrusion is defined as an event that tries to compromise the confidentiality, integrity and availability of a resource which are the basic information security properties. A network should be examined regularly for intrusion attempts in order to secure an organization completely. The mission of an Intrusion Detection System (IDS) is to detect both a non-authorized intruder who tries to access the system and an authorized user who misuses the system resources. IDSs are the security tools that examine local audit data and network traffic in order to find the deviations from normal computer behaviors and predefined traffic patterns.

Usually, an intruder gets access to one user first by misusing the software vulnerabilities, later tries to login other network hosts via the preempted host. The objective of a Denial of Service (DoS) attack is preventing the victim system to serve its normal services. In Distributed Denial of Service attack (DDoS), a victim is attacked from multiple sources simultaneously. These attacks usually use the preempted computers which are ready to be misused for launching an attack. These zombie computers participate in the intermediate phase of the attack.

The objective of this dissertation is to detect pre-DDoS attacks in the intermediate level distributedly using mobile agents and nature inspired algorithms and inform the security managers before the attack succeeds. After fulfilling the construction of the distributed structure and generation of an anomaly-based IDS with the help of nature inspired algorithms seperately, these two systems are combined together to build an IDS which can detect new attacks with lower false positive rates, have the capability to update itself adaptively, highly secure since it can work independently from the central unit, lightens the network load since it makes the intrusion detection using mobile agents, and detects succesively the DDoS attacks in the intermediate phase which is called “botnet” as they are not aware of being misused.

In the first part, six different distributed intrusion detection method is designed in order to detect the DDoS attacks in the MIT DARPA LLS_DDOS 1.0 dataset and tested in a simulated-real time environment respecting the occurring time of the attacks. Mobile agents are used in these signature-based IDS methods except the first one. In Method_1, there is a centralized distributed IDS without mobile agents in which a distributed intrusion decision is given and the security manager is informed if there comes an intrusion-suspect message from different hosts in an hour with same source IP. In Method_2, the decision of distributed intrusion is given by Mobile Agents which are created by the MainAgent in the above stated conditions and which visit only the two hosts that are in their lists to examine the related data on their origin. In Method_3, MainAgent doesn’t wait the number of required messages for creating a Mobile Agent to be two, and immediately dispatches the related agents to the network. In Method_4, Mobile Agents are created by Static Agents of hosts

coordinated with the center. In Method_5, Mobile Agents are created and sent to the system by the Static Agents which detect the intrusion-suspects without coordinating with the MainAgent and other hosts. This is a fully-distributed method since MainAgent is never used. Mobile Agents travel around the network in a random order in order to give the decision of distributed intrusion. In Method_6, a high reliable system with short detection time is developed by combining the short mean detection time advantage of Method_2 and fully-distributed architecture of Method_5. This system normally works in mode-1 (Method_2), but shifts to mode-2 (Method_5) when the central unit is not available.

It is aimed to detect first three phases of DDoS attacks that are in the dataset accurately and fast, and warn the security manager to take necessary precautions before the occurrence of last two phases. As a result of twenty runs, mean detection times, network load and reliability properties of each methods are evaluated individually. Method_4 is considered to be the best method for this part of the study, since it is fully-independet from the central unit and causes low network load.

In the second phase, artificial immune system is used as the method of anomaly-based intrusion detection because of the similarity between the IDS architecture and biological immune system. New improvements are added to the jREMISA study which is a multiobjective evolutionary algorithm inspired artificial immune system, in order to obtain better true and false positive rates while detecting the DDoS attacks that are in the MIT DARPA LLS_DDOS 1.0 dataset. These improvements can be stated as addition of r-continuous evaluation method, changes in Negative Selection and Clonal Selection parts, re-defination of the second objective while keeping the overall concept same.

Three different tests are performed with different configurations of the parameters of affinity threshold values, r-continuous values and the size of the primary population in order to find the best parameter group that will give better true and false positive results over the improved-jREMISA. At last, original and improved-jREMISA are compared using the predefined best parameter group. 100% true positive rate and 0% false positive rate of the improved algorithm is a noteworthy success for an anomaly intrusion detection system.

Later, some experiments are performed using different combinations of 1999 DARPA IDS dataset days which is commonly used in literature in order to compare the performance of the improved-jREMISA with other similar studies. The success of approximately 100% true positive values and 0% false positive values is obtained in these experiments. It is observed that imroved-jREMISA had better TP nad FP values than all of the others when compared with the studies that are tested with the same dataset.

Lastly, the target system of this dissertation is created by combining these two seperately developed structures. In the distributed environment, the intrusion-suspect alarms of signature-based IDS of Snort are substituted by the alarms which are generated by anomaly-based IDS of the improved-jREMISA. Method_2, Method_3 and Method_4 of signature-based system are decided not to be used in anomaly-based system because of their functional similarity to other methods. Other three methods which were developed according to the signature-based controls are reset according to the anomaly-based detection structure. Mobile Agents increase the DDoS state of the hosts where they find a similar anomly-type intrusion-suspect packet while questioning the alarm files of the them, and controls everytime whether

a malicuous software is uploaded to the hosts. Security manager is informed about these attack phases via the AlarmAgent which is assumed to be stated in a secure area.

The tests with 1.0 version and more sophisticated 2.0.2 version of MIT DARPA 2000 LLS_DDOS dataset are resulted successfully. A distributed IDS is developed, which alerts the security manager before the DDoS attack launch, able to work independently from the central unit if necessary and not causes more than enough network load. Although there are studies about "distributed detection of DDoS attacks", "distributed detection of intrusions using mobile agents" and "anonaly-based IDS"; detection of pre-DDoS distributed preparatory traffic on the intermediate phase which is composed of zombie computers using an anomaly-based IDS which works with high correct detection rates is performed first time in this study.

1. GİRİŞ

Bilgi güvenliği, hassas elektronik bilgilere erişimin kontrol edilmesi ve böylece sadece yetkisi olanların erişimine izin verilmesidir. Bu kontrol sırasında kullanıcıların istekleri ile veri gizliliği ve bütünlüğü ihtiyacını dengelemek gerekmektedir. Çalışanların bir işyeri ağına, evlerinden veya seyahatteyken uzaktan erişimine izin vermek, ağın kalite değerini ve çalışanların verimliliğini artırabilir. Ama ne yazık ki, uzaktan erişim, aynı zamanda ağın güvenliğini korumayı zorlaştıracak çok sayıda zayıflıklara da neden olabilmektedir [1].

Bilgi güvenliğinin gizlilik, bütünlük ve verinin kullanılabilirliği olmak üzere üç temel amacı vardır. Gizlilik, bilginin bir güvenlik politikasına uygun olarak saklanması ve açıklanması; bütünlük, bilginin korunması, bozulmaması ve sistemin düzgün çalışması; kullanılabilirlik, sistem ünitelerinin istenildiğinde kullanılmaya hazır olması anlamına gelmektedir. Sızma, bir kaynağın gizlilik, bütünlük ve kullanılabilirliğini tehlikeye sokmaya çalışan olay olarak tanımlanmaktadır. Saldırdıkları makinelerin yetkilendirilmemiş kullanıcıları olan dış sızıcılar ve sisteme bazı kısıtlar dahilinde erişme izni olan iç sızıcılar olmak üzere iki türlü sızıcı vardır [2].

Bilgisayar güvenliğinde savunmanın ilk hattı olarak kullanıcı asıllama, veri şifreleme, program hatalarından sakınma ve ateş duvarları gibi geleneksel korunma teknikleri kullanılmaktadır. Fakat, bir şifre zayıfsa ve saldırı tehlikesi altındaysa kullanıcı asıllaması yetkisiz kullanımı engelleyemez. Ateş duvarları, konfigürasyon hatalarına ve belirsiz güvenlik politikalarına karşı savunmasızdır; kötü niyetli gezgin kodlara, iç saldırılara ve güvenliksiz modemlere karşı korumaları yoktur.

Gerekli yerlere erişim-kontrol ölçümleri koymak kadar, bir sızıcının bu noktaları delerek geçip geçmediğini doğrulamak da önemlidir. Bir organizasyonu tam koruma altına almak için, ağı düzenli olarak sızma girişimleri için gözlemlemek gerekmektedir. Sızma Belirleme Sistemi (SBS), bir hırsızlık alarmına benzemektedir. Bir sızıcı, hırsızlık alarmına takıldığında, ev sahipleri, komşular ve kanun koruyucular bu kanun dışı giriş eylemi hakkında uyarılmaktadırlar. Aynı durum SBS

için de geçerlidir. Bir SBS'in görevi hem sisteme erişmeye çalışan yetkisiz bir sızıcıyı, hem de sistem kaynaklarını kötüye kullanan yetkili bir kullanıcıyı belirlemektir. SBS'in en ünlü özelliği koruyuculuk kabiliyetinden ziyade belirleyicilik kabiliyetidir. Proaktif değil, reaktif davranışlara sahiptir [1].

1.1 Problemin İncelenmesi

İnternete bağlı olan bilgisayar ağları, sürekli olarak çok sayıda ve çeşitli siber suçlara maruz kalmaktadırlar. Kötü niyetli bir internet kullanıcısı, diğer bilgisayarlarda yer alan özel bilgilere erişebilir, değiştirebilir, silebilir veya bazı bilgisayar hizmetlerini diğer kullanıcılar için kullanılamaz hale getirebilir. Günümüzdeki bilgisayar ağlarının altyapısı o kadar büyük ve karışıktır ki bu ağları tamamen güvenli hale getirmek neredeyse imkansızdır. Bundan dolayı, bilgisayar kaynaklarının gizliliği, bütünlüğü ve kullanılabilirliği saldırıya uğradığı zaman etkili biçimde bunu belirlemek ve cevap verebilmek için bir SBS'ne ihtiyaç vardır.

Genelde, Hizmetin Engellenmesi Saldırısında (DoS) olduğu gibi, bir saldırıcı öncelikle yazılım açıklarını suiistimal ederek tek bir kullanıcıya erişim sağlar, daha sonra önceden ele geçirdiği konakçı vasıtasıyla ağdaki diğer konakçılara girmeyi dener [3]. DoS saldırısının amacı hedef sistemin normalde sunduğu hizmetleri sunamaz hale getirmektir. Dağıtık Hizmetin Reddedilmesi Saldırısında (DDoS), aynı anda çok sayıda kaynak tarafından tek bir hedefe saldırı düzenlenir. DDoS saldırıları genelde, önceden başkasının kendilerini bir saldırı başlatmak için kullanabilmesi amacıyla ele geçirilmiş olan bilgisayarları kullanırlar. Bu zombi bilgisayarlar saldırının orta katmanında rol alırlar [4].

1.2 Tezin Amacı ve Özgün Katkıları

Dağıtık Sızma Belirleme Sistemlerinde (dSBS) amaç, ağdaki bilgisayarlar ve ağ teker teker incelendiğinde anlaşılamayan ancak, bu veriler birleştirildiğinde ortaya çıkabilen sızmaları belirlemektir. Gezgin etmenler kullanılarak verilerin toplanması ve analiz edilmesi işlemlerinin yükü ve riski merkezden alınıp ağa dağıtılması düşünülmüştür. Böylece ağdaki veri yükü artmayacak ve merkezin bir saldırıya uğraması durumunda tüm sistemin çalışamaz duruma gelmesi engellenecektir. Doğadan Esinlenen Algoritmalar kullanılarak SBS'in duyarlılığının yeni tip

sızmalara adapte olabilmesi amaçlanmıştır. Kullanılan SBS'ler DDoS saldırılarını belirlemek için ağ trafiğini inceliyor olsa da konakçı-temelli SBS'lerdir, çünkü inceledikleri trafik ayrı ayrı konakçılara ait olan trafiktir.

Bu konuda şimdiye kadar yapılan çalışmalarda, dağıtık veri toplama ve analiz etme yapısı dinamik öğrenme yöntemleriyle beraber kullanılarak istenilen etkinlik seviyesinde henüz uygulanamamıştır. Gezgin etmenleri yaratan ve yönlendiren merkezi birimlerin kendilerinin saldırılara karşı korumalı hale getirilmesi veya merkezi hiçbir birimin olmaması konuları ayrı ayrı çalışmalarda ele alınmıştır. Bu tezde, gezgin etmenler ve doğadan esinlenen sezgiseller kullanarak dağıtık bilgisayar güvenliğinin sağlanması amaçlanmıştır. Özellikle DDoS saldırılarının en hızlı ve doğru olarak belirlenmesi, SBS'inin kendisinin de bu tür saldırılara karşı korunabilmesi öngörülmüştür.

Bu çalışmada, DDoS öncesi saldırıları orta katmanda belirleyebilecek dağıtık sızma belirleme yöntemleri önerilmektedir. Birincisi dışındaki diğer yöntemlerde gezgin etmenler kullanılmaktadır. İlk bölümde, bütün yöntemler imza-temelli bir SBS olan Snort tarafından oluşturulan alarmları kullanırken, ikinci bölümde anomali-temelli bir SBS olarak geliştirilmiş-jREMISA'nın oluşturduğu alarmları kullanmaktadırlar. Daha gerçekçi sonuçlar alabilmek için yeni bir simule-gerçek zamanlı test ortamı kullanılmıştır.

SBS mimarisi ve antijenleri belirleyen parallel ve dağıtık adaptif bir sistem olan Biyolojik Bağışıklık Sistemi arasındaki benzerlikten dolayı, bir anomali-temelli SBS yöntemi olarak Yapay Bağışıklık Sistemini (AIS) kullanıldı. AIS-temelli bir SBS, antijen belirleyici popülasyonu eğiterek ağ trafiğini kendinden ve kendinden-olmayan olarak sınıflandırır. Çok-amaçlı Evrimsel Algoritmadan (MOEA) esinlenen bir AIS olan jREMISA [5], daha önce gerçekleştirilmiş olan bir çalışmadır. Bu tez çalışmasında, MIT DARPA 2000 LLS_DDOS 1.0 ve 2.0.2 ile 1999 verisetindeki DDoS saldırılarını belirlerken daha iyi doğru ve yanlış pozitif oranları elde edebilmek için jREMISA çalışması daha da geliştirildi.

Sonuç olarak, DDoS saldırılarını orta katmanda doğru ve etkin olarak belirleyen ve gerçeklenmesinin engellenmesi için gerekli uyarıları zamanında verdiren, sürekli öğrenme yeteneğine sahip, gezgin etmen yapısını kullanarak ağdaki yükü hafifleten ve sistemin güvenirliğini artıran yeni bir dSBS ortaya konmuştur.

2. LİTERATÜR TARAMASI

2.1 Kavramlar

Bu bölümde, tez çalışmasının temelini oluşturan ana kavramlardan olan SBS'leri, gezgin etmenler, dağıtık saldırılar ve doğadan esinlenen algoritmalar hakkında detaylı bilgi verilmektedir. Deneylerde kullanıldıkları için, SBS konusu içerisinde anomal-temelli sistemler, dSBS ve Snort, gezgin etmenler konusu içerisinde Jade, dağıtık saldırılar konusu içerisinde DDoS ve doğadan esinlenen algoritmalar konusu içerisinde Evrimsel Algoritmalar (EA), Çok-amaçlı Evrimsel Algoritmalar (MOEA) ve Yapay Bağışıklık Sistemi konuları ayrıca ele alınmaktadır.

2.1.1 Sızma belirleme sistemleri

Sızma belirleme sistemlerinin temel amacı, bilgisayar sistemlerinin iç ve dış sızıcılar tarafından hatalı, yetkisiz ve kötü amaçlı kullanımını belirlemektir. Bir SBS'nin işlevselliği mantıksal olarak üç bileşen ile gruplandırılabilir: algılayıcılar, çözümleyiciler ve bir kullanıcı arayüzü.

1. Algılayıcılar, veri toplamakla sorumludurlar. Bir sızma izi taşıyan herhangi bir sistem parçası algılayıcı için girdi olabilir. Ağ paketleri, log dosyaları ve sistem çağrıları örnek algılayıcı girdileridir. Algılayıcılar bu bilgileri toplar ve çözümleyiciye iletirler.
2. Çözümleyiciler, bir veya daha fazla algılayıcıdan veya diğer çözümleyicilerden girdi alırlar. Çözümleyici, bir sızmanın gerçekleşip gerçekleşmediğini değerlendirmekle sorumludur. Bu bileşenin çıktısı, sızmanın gerçekleştiğine dair bir işarettir. Çözümleyici, sızma sonucunda neler yapılabileceği hakkında yol gösterici de olabilir.
3. Kullanıcı arayüzü, bir kullanıcının sistemin çıktısını görebilmesini veya sistemin davranışlarını kontrol etmesini sağlar. Bazı sistemlerde, kullanıcı arayüzü, yönetici veya konsol bileşenine denk gelmektedir.

Bu üç önemli bileşene ilave olarak, SBS, bilinen zayıflıkları olan ve sızıcıya görünür olacak şekilde tasarlanan bir çekici-alan (honey pot) da içerebilir. Çekici-alanlar sayesinde sızmalar kontrollü bir ortamda belirlenebilirler [1].

SBS'leri sınıflandırma yollarından iki tanesi analiz yaklaşımı ve SBS'in yerleşimidir. Analiz yaklaşımına göre iki sınıf vardır: hatalı kullanımın belirlenmesi ve anomalinin belirlenmesi.

Hatalı kullanımın belirlenmesi yaklaşımında, ağ ve sistem faaliyetleri bir tür örüntü eşleme (pattern matching) kullanılarak bilinen hatalı kullanımları bulmak için incelenir. Mesela, "cmd.exe" dosyası ile ilgili olarak yapılan bir http isteği bir saldırı belirtisi olabilir [6]. Aksine, anomali belirleme yaklaşımında, kararlar normal ağ veya sistem davranışlarına dayandırılır. Kullanıcının davranışları gözlemlenir ve normal davranıştan olan az bir sapma, sızma olarak belirlenir. Sistem faaliyetlerinden türetilen ölçütlerden oluşan bir model oluşturulur. Bu ölçütler, ortalama Merkezi İşlemci Birimi (MİB) yükü, dakikadaki ağ bağlantı sayısı, kullanıcı başına düşen işlem sayısı gibi sistem parametrelerinden elde edilir [7]. Mesela, normalde pasif bir web sunucusunun çok fazla sayıda adrese bağlantı açmaya çalışması bir solucan bulaşmasının belirtisi olabilir [6].

Hatalı kullanımın belirlenmesi temelli bir sistemin yapısal düzenleme ihtiyacı, genelde anomali-temelli bir sistemden daha azdır; çünkü anomali-temelli sistemin daha fazla veri toplanmasına, analizine ve güncellenmesine ihtiyacı vardır. Sistemin bilinen ve beklenilen davranışlarının geniş bir tanımı gerektiği için, anomali-temelli sistemleri düzenlemek daha zordur. Bazı durumlarda sistematik destek sağlanabilmektedir, fakat sistemin geliştirilmesi çok zaman alıcıdır ve kullanılan verilerin kesin olması gerekmektedir [1].

Normal bilgisayar veya ağ verisi az miktarda saldırı verisi içerdiği için, düşük doğrulanmamış uyarı (yanlış pozitif) oranı SBS'ler için çok kritik öneme sahiptir. Bir ateş duvarı içerisine yerleştirilmiş olan SBS'in bir günde 1000000 ağ paketi işlediğini ve bunların sadece 10 tanesinin gerçek saldırı olduğunu varsayalım. Bu SBS'nin yanlış pozitif oranı %1 ve belirleme oranı %90 olsun. Gün içerisinde 10000 tanesi yanlış pozitif olan 9 tanesi de doğru pozitif olan toplam 10009 adet alarm üretilecektir. Bu 9 adet doğru pozitifi bulmanın ne kadar çok zaman alacağı aşikardır ve sızma belirlemenin güçlüğü görülebilmektedir [8].

Her iki yaklaşımın da güçlü ve zayıf yönleri vardır. Kritik nokta, yanlış pozitif gerçekleşmesini azaltırken aynı anda gerçek uyarıları (doğru pozitif) çoğaltmaktır. Hatalı kullanım temelli sistemlerin, genelde yanlış pozitif oranları düşüktür; fakat yeni saldırıları tanıyamadıkları için yanlış-negatif oranları yüksektir. Anomali-temelli sistemler ise, yeni saldırıları belirleyebilirler ama yüksek sayıda yanlış-pozitif üretirler. Bu durum, mevcut anomali-temelli sistemlerin, gerçek dünyadaki bilgisayar ağ ve sistem kullanımının zamanla değiştiği gerçeğiyle yeterince baş edememelerinden kaynaklanmaktadır. Dolayısıyla, herhangi bir normal davranış profili de dinamik olmak zorundadır [9].

SBS'lerin yerleşimine gore de, SBS'ler konakçı–temelli ve ağ–temelli olarak ikiye ayrılırlar. Konakçı–temelli sistemler, izlenilmek istenilen konakçılara kurulurlar ve log dosyaları, konakçıya ve konakçıdan olan ağ trafiği veya konakçıda koşan prosesler hakkında bilgiler gibi konakçının işlemlerini ilgilendiren veriler toplarlar. Ağ–temelli sistemler ise, korunacak olan konakçıların olduğu ağdaki ağ trafiğini izlerler ve genelde algılayıcı denilen ayrı bir makinede koşturulurlar.

İki sistemin de avantaj ve dezavantajları vardır. Konakçı–temelli sistemler, bir saldırı denemesinin gerçekten başarılı olup olmadığının kararını verebilir ve yerel saldırıları, ayrıcalık artırma saldırılarını, şifreli saldırıları belirleyebilir; ama ağdaki birden fazla hedefe yapılan saldırıları belirleyemezler. Ancak, korunma ihtiyacı olan konakçı sayısı fazla olursa, bu sistemin yerleştirilmesi ve yönetilmesi zor olur. Ağ–temelli sistemler, çok sayıda konakçıyı nispeten daha düşük maliyetle izleyebilirler ve birden fazla konakçıya yapılan saldırıları tanıyabilirler. Fakat, bir saldırı denemesinin gerçekten başarılı olup olmadığını belirleyemezler, yerel ve şifreli saldırılarla baş edemezler. Birden fazla kaynaktan yapılan saldırılara karşı korunmak için, konakçı ve ağ–temelli sistemleri birleştiren melez sistemler kullanmak en koruyucu tercih olacaktır [9].

2.1.1.1 Anomali-temelli SBS

Anomali-temelli sistemler çoğunlukla istatistiksel veya makine öğrenmesi teknikleri kullanılarak inşa edilir.

İstatistiksel modeller

İstatistiksel modeller, yeni gözlemin önceki gözlemlere göre anormal olup olmadığını değerlendirmek için, her kullanıcı için metriklerden ve önceki

gözlemlerden oluşturulmuş olan istatistiksel normlardan sapmaları belirler. Anomali belirleme sistemlerinde kullanılan üç adet istatistiksel model şunlardır: [10]

1. İşlemsel (operational) model: Anormalliğin, yeni gözlemlerin sabit limitlerle karşılaştırılması sonucunda elde edileceği varsayımına dayanır. Bu limitler aynı tipteki verilerin önceki gözlemlerinden elde edilir. Örneğin, kısa bir zaman dilimindeki şifre girme başarısızlıklarının sayısı 5'ten fazlaysa, zorla girme denemesi yapıldığı belirlenir.
2. Ortalama ve standart sapma modeli: Ortalama, standart sapma ve güven aralığı (confidence interval) denklemlerini temel alır. Eğer yeni yapılan gözlem, belirli parametreler için ortalamadan olan d kadar standart sapmayı içeren güvenlik aralığının dışında kalırsa, anormal olarak tanımlanır.
3. Zaman serileri modeli: Olay sayacı veya kaynak ölçeğiyle beraber zaman aralığı sayacı da kullanan model, gözlemlerin değerleri kadar sırası ve gerçekleşmeleri arasında geçen süreyi de değerlendirir. Gözlenen bir olayın o zaman diliminde gerçekleşme ihtimali çok düşükse, o olay anormaldir. Örneğin, iki oturum açma hareketi arasında geçen zamanlar $x_{1:2}$=.1, $x_{2:3}$=.07, $x_{3:4}$=.05 saniye olsun. Bir sonraki gözlemin değeri, $x_{4:5}$, güven aralığının dışında kalırsa, anormal olarak belirlenir.

İstatistiksel modelin, kullanıcı ve sistem davranışlarını "audit log" kayıtlarından öğrenebilmesi gibi avantajları yanında bazı sınırları da vardır:

- Saf istatistiksel sızma belirleme sistemleri, anormal davranışların normal olarak değerlendirildiği bir aşamaya yavaş yavaş getirilebilir. Gözlendiklerinin farkında olan sızıcılar sistemi bu şekilde eğitebilirler.
- Uygun eşik değerleri bulmak zor olduğu için yüksek yanlış pozitif oranlarına sahiptirler.
- İstatistiksel ölçümler, olayların oluş sırasına duyarlı değildir: "finger" komutunu takip eden "login" komutu bu duruma bir örnek olabilir. Birbirlerini takip eden olaylar arasındaki ilişkilerin gösterdiği sızmaları belirlemeyebilirler.

Makine Öğrenmesi modelleri

Sızmaların ve saldırıların karmaşıklığı arttıkça ağ güvenliği sorunlarını çözmek için Kümelere Ayırma (Clustering), Dıştakilerin Belirlenmesi (Outlier Detection), Destek Vektörü Makinesi (Support Vector Machine-SVM), Gizlenmiş Markov Modelleri (Hidden Markov Models-HMM), Yapay Sinir Ağları (Artificial Neural Networks-ANN) gibi makine öğrenmesi ve veri madenciliği modelleri daha çok kullanılmaya başlanmıştır. Kümelere Ayırma ve Dıştakilerin Belirlenmesi yöntemlerinin etkili olabilmesi için "audit" verisi içindeki anormal veri oranının düşük olması gerekmektedir, bu nedenle, yoğun DDoS ve yoklama saldırılarında belirleme oranları çok düşük olmaktadır. SVM, HMM ve ANN, daha yüksek sınıflandırma oranlarına sahiptirler; fakat normal kullanıcı profillerini tarif eden algoritmalarının karmaşıklığı gerçek uygulamalardaki etkinliklerine gölge düşürmektedir [10].

Yapay Sinir Ağları

ANN'ler, girdi-çıktı vektörleri arasındaki ilişkiyi öğrenen ve bunları, mantıklı bir şekilde yeni girdi-çıktı vektörleri elde etmek için genelleyen algoritmalardır. ANN'in bazı avantajları aşağıda anlatılmıştır: [10]

- İstatistiksel yöntemler, normal profilden olan sapmaların Gauss dağılımında olduğu gibi, kullanıcı davranışlarının dağılımları hakkında bazı varsayımlara temel alır. Bu varsayımlar doğru olmayabilir ve yüksek yanlış pozitif oranına neden olabilir. ANN'in bu varsayımlara ihtiyacı yoktur.
- Kullanılan "audit" verisi için geçersiz olan varsayımları kaldırdıktan sonra, bunları değişik davranış karakteristiğine sahip yeni bir kullanıcı topluluğuna uygularken algoritmaların değiştirilmesi gerekmektedir. İstatistiksel algoritmaları yeniden yapılandırmanın ve onları gerçekleyen yazılımları yeniden oluşturmanın maliyeti yüksektir. Sinir ağları simülatörlerini yeni kullanıcılara göre ayarlamak daha kolaydır.
- Binlerce kullanıcı içeren büyük kullanıcı topluluklarında, davranışlardaki benzerliklere göre bireyleri gruplara ayırmak için ANN kullanılabilir. Böylece, her bir birey için ayrı ayrı profil yerine grup profilleri saklanır.

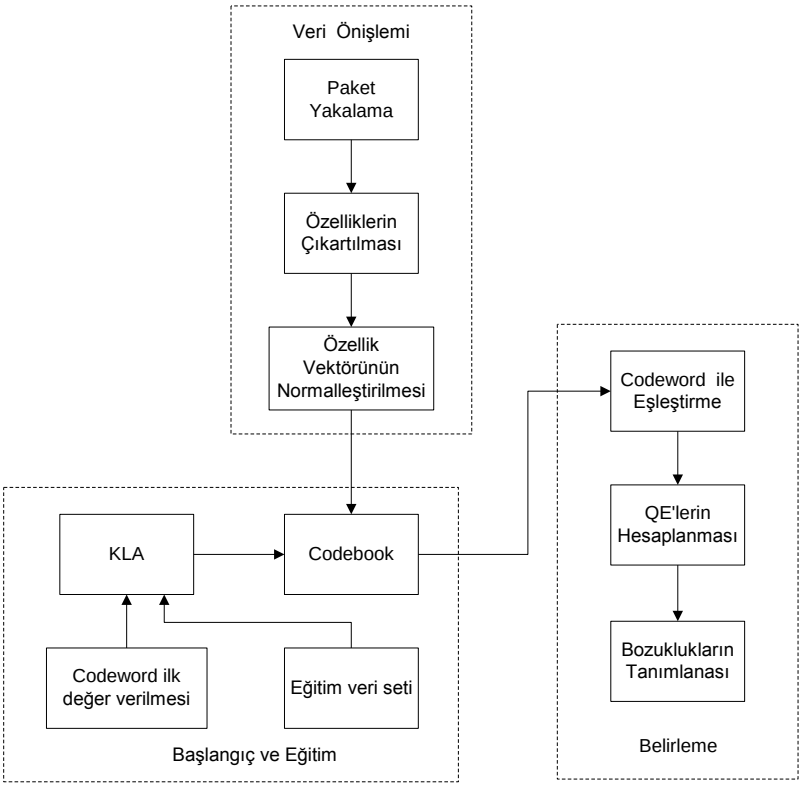
ANN, sızma belirleme için çok faydalı olsa da, tamamen istatistiksel modellerin yerini alamaz. ANN'in temel kullanım alanı sistemdeki kullanıcıların davranışlarını öğrenmek ve tahmin etmektir.

Vektör büyüklüğünün ölçümü (Vector Quantization)

Vektör Büyüklüğünün Ölçümü, bir veri setinin kümelere ayrıştırılması yaklaşımıdır. Kullanıcı profili analizi yaparken, özellik vektörlerinin benzerliklerini karşılaştırarak trafik özellikleri vektör uzayını gruplara ayırır. Bütün profillerin özü çıkartılır ve "codebook" denilen yapıda saklanır [11].

"Codebook", normal ağ trafiğinin bir algoritma kullanılarak eğitilmesiyle oluşturulur. Benzer davranışların minimum Euclid mesafelerine göre gruplandırıldığı normal ağ trafik davranışları sözlüğü gibi düşünülebilir.

Belirleme aşamasında, girdi özellik vektörleri Şekil 2.1'de gösterildiği gibi "codebook" ile işleme sokulur. En iyi eşleştiği "codeword"ün bulunması için en yakın komşunun aranması yöntemi kullanılır. Girdi uzayındaki kısa dönemli davranışlar ile "codebook'taki uzun dönem davranışlar arasındaki benzerlikleri ölçmek için en yakın komşuya olan Euclid mesafesi olarak tanımlanan Büyüklük Ölçümü Hataları (Quantization Errors) kullanılır.



Şekil 2.1 : Vector Büyüklüğünün Ölçümü [11].

Saldırı tanıma kabiliyetinin geliştirilmesi için etkili belirleme modelleri ve algoritmalarının yanında, doğru özellik vektörü oluşturulma metodlarının kullanılması gerekmektedir. Trafik özellikleri normal trafik profilini anormal profilden maksimum derecede ayıracak şekilde belirlenmelidir. TCP trafik akımının özellik vektörü aşağıdaki Çizelge 2.1'de yer almaktadır.

Çizelge 2.1 : TCP akımı özellik vektörü [11].

Özelliğin Niteliği	Açıklama
SrcIP	Kaynak IP adresi
DestIP	Hedef IP adresi
SrcPort	Kaynak portu
DestPort	Hedef portu
PktSize	TCP akımındaki ortalama paket büyüklüğü
SrcBytes	Kaynaktan gelen byte sayısı
DestBytes	Hedeften gelen byte sayısı
FlowState	TCP akımının kapalı durumu
Fre_SrcIP	Zaman-penceresi içinde kalan kaynak IP sıklığı
Fre_DestIP	Zaman-penceresi içinde kalan hedef IP sıklığı

Özellik niteliklerinin en önemlilerinden olan FlowState hesaplaması şu şekilde (2.1) yapılmaktadır: [11]

$$FlowState = Toplam(Bayrak_0, Bayrak_1, ..., Bayrak_8) = \sum_{i=0} Bayrak_i \cdot 2^i$$
$$= RST_{active} \cdot 2^0 + RST_{passive} \cdot 2^1 + SYN \cdot 2^2 + ACK_{syn} \cdot 2^3 + ACK \cdot 2^4 \quad \textbf{(2.1)}$$
$$+ FIN \cdot 2^5 + ACK_{fin} \cdot 2^6 + FIN' \cdot 2^7 + ACK'_{fin} \cdot 2^8$$

Normal kapanışlı bir TCP bağlantısında Toplam=$(111111100)_2$= $(508)_{10}$ olacaktır. Bir SYN yoklama saldırısının iki durumu incelenirse:

- Hedef port açık değildir ve hedef bilgisayar $RST_{passive}$ ile cevap verir:

  Toplam= $RST_{passive} . 2^1 + SYN . 2^2 = 1x2 + 1x4 = 6$

- Hedef port açıktır ve hedef bilgisayar ACK_{syn} ile cevap verir:

  Toplam= $RST_{active} . 2^0 + SYN . 2^2 + ACK_{syn} . 2^3 = 1 + 1x4 + 1x8 = 13$

6 ve 13 rakamları anormal durumu ifade ederler.

2.1.1.2 Dağıtık Sızma Belirleme Sistemleri

Saldırıcı, sızma faaliyetlerinin seviyelerini her bir konakçı üzerinde bulunan sızma izleme programları tarafından kabul edilebilir düzeyde tutarak, bir ağdaki birden

fazla sisteme eş zamanlı olarak sızabilir. Fakat, ayrı ayrı konakçılardaki saldırı faaliyetlerinin seviyelerinin toplamından oluşan toplam sızma faaliyet seviyesi, bir alarm durumu oluşturacak kadar yüksek olabilir. Bir ağda, izlenen bütün konakçılardan sızma faaliyeti verilerini toparlayan ve birbirleriyle ilişkilendiren SBS türüne Dağıtık SBS denilir [3].

Bir dSBS, geniş bir ağ üzerinde, birbirleriyle veya gelişmiş ağ izleme ve olay analizi yapabilen bir merkezi sunucu ile haberleşen çok sayıda SBS içerir. Birbirleriyle yardımlaşan etmenlerin ağa dağılmış olmaları sayesinde analistler, ağ operatörleri ve güvenlik personeli kendi ağlarında neler olup bittiğini bir bütün olarak geniş çerçeveden görebilme imkanına sahip olmaktadırlar.

dSBS, ağ ortamında bağımsızca hareket eden küçük prosesler (etmenler) barındırır. Bu etmenler, içine kondukları ortamda gezerken uğradıkları sistemin davranışlarını ve hangi kullanıcıların o sisteme giriş yaptıklarını gözlemlemek, birbirleriyle mesajlar aracılığıyla işbirliği yapmak, bir olay şüpheli olarak değerlendirildiğinde birbirlerini haberdar etmek, karşı saldırılar yapmak amacıyla geliştirilirler.

Basit saldırıların izleri, tek bir log dosyasında bile görülebilmekte veya basit bir ağ arayüz kartından bile izlenebilmektedir. Halbuki, iyi bilgisayar korsanları, yalnızca tek bir konakçı üzerine yoğunlaşmazlar, eylemlerini çok sayıda makine üzerine dağıtarak saklamaya çalışırlar. Bu eylemler, tek başlarına değerlendirildiğinde masum gibi görünseler de topluca değerlendirildiklerinde bir saldırının parçaları oldukları görülebilmektedir. Bu yüzden, farklı kaynaklardan izleme verisi toplamak ve bunları ilişkilendirmek gerekmektedir [12].

Bilgileri birleştirme, bu gelişmiş analiz yönteminin temel bileşenidir. Benzer veya ilişkili verilerin toplanması sayesinde, analistler bir saldırının aktif ağ keşfinden en son saldırı aşamasına kadar olan farklı aşamalardan nasıl geçtiğini kolaylıkla görebilmektedir. Olay çözümleyiciler, saldırıcının hangi zaman aralığında çalıştığını ve çok sayıda birbiriyle yardımlaşan saldırıcının varlığını anlamak için ağa yapılan diğer saldırı denemelerini ilişkilendirmek imkanına sahip olmaktadırlar. En yaygın bilgi birleştirme yöntemleri, saldırıcı IP numarası, hedef port numarası, etmen ID numarası, tarih, zaman, protokol vaya saldırı tipine göre yapılmaktadır.

Mevcut SBS'lerin çoğu, dağıtık veri toplamalarına rağmen veri işlemelerini merkezi olarak yapmaktadırlar. Bu durum, sistemlerin ölçeklenebilirliklerini, kolay

ayarlanabilirliklerini ve hata toleranslarını sınırlamaktadır. Merkezi ünitenin çökmesi, ilişkilendirme işlemini tamamen inaktif etmekte ve SBS'ini körleştirmektedir. Bu merkezi düğümün işlem kapasitesi de belirli bir zaman diliminde ilgilenebileceği olay sayısını sınırlandırmaktadır. Çok sayıda algılayıcı mesajlarını merkezi konakçıya gönderdiğinde oluşacak olan birikme, sistemin reaksiyon süresini artıracak ve hatta veri kaybına neden olacaktır.

dSBS, etmenlerin analistlere sağladığı geniş görüş sayesinde, olay analistlerine diğer tekil SBS'lerden daha fazla avantajlar sunmaktadır. Bu avantajlardan bir tanesi, zaman dilimleri veya kıtalarla birbirinden ayrılan coğrafik bölgelerde kurulu olan geniş ağlardaki saldırı paternlerini belirleme kabiliyetidir. Böylece, iyi planlanmış ve koordineli saldırılar erkenden belirlenebilmekte ve güvenlik personeli tarafından hedeflenen sistemlerin güvenliğinin sağlanmasına ve saldırılan IP'lerin erişimine izin verilmemesine imkan tanınmaktadır. Diğer bir avantajı ise, geniş bir ağda yoluna devam eden bir internet solucanının erken teşhisine imkan tanımasıdır. Bu bilgi, solucanın bulaştığı sistemleri belirlemek, temizlemek ve solucanın ağdaki yayılmasını durdurmak için kullanılabilir. Sonuçta, ortaya çıkabilecek maddi kayıplar azaltılabilecektir.

İkinci büyük avantajı da, fiziksel mesafelerden dolayı önceden birden fazla olay analistinin yaptığı işleri artık tek bir olay analistinin yapabilecek olmasıdır. Bu durumda, organizasyonun ofislerinin bulunduğu farklı coğrafik bölgelerdeki olay analistlerine ödeme yapma ihtiyacı da kalmayacaktır.

Bahsedilmesi gereken diğer bir konu da, şirket ağı içerisindeki kızgın, küskün veya sıkılmış olan çalışanlar tarafından yapılan saldırılardır. Merkezi analiz sunucusu şirketin DHCP (Dynamic Host Configuration Protocol) veya RADIUS (Remote Authentication Dial-in User Service) sunucularıyla bağlanarak, kurum içerisinden saldırı düzenleyen kişiler ve yapmaya çalıştıkları eylemler takip edilebilmekte ve faillere karşı delil toplanabilmektedir [12].

2.1.1.3 Snort

Snort [13], trafiği analiz etmek için kurallar ve önişlemciler kullanan açık-kaynak, ücretsiz, imza-temelli bir ağ SBS'dir. Marty Roesch tarafından 1990 yılında geliştirilmiştir. Kurallar, tek bir paketi incelemek amacıyla basit ve esnek şekilde imzaların oluşturulmasını içerir. Önişlemciler, parçaların birleştirilmesi

(defragmantasyon), port taraması belirlemesi, web trafiğinin normalleştirilmesi gibi kuralların tek başına yapamayacağı detaylı incelemeleri yapar. Kurallar, paketleri tek tek incelerken, önişlemciler birçok paketi birden değerlendirebilirler.

Snort, topladığı ağ trafiğini ekrana yazdırmaktan, önceden ayarlanmış imzalardan oluşan kurallarla ağ trafiğini karşılaştırdığı ağ SBS moduna kadar değişik modlarda çalışabilmektedir. Fakat, en yaygın olarak ağ SBS modunda çalıştırılır.

Snort, hem başlık hem de yük inceleme yöntemlerini destekler. Şüpheli bir paketin tek bir kural içerisinde detaylı olarak tarif edilmesine imkan tanır. Bu esneklik sayesinde, yanlış pozitif oranını en aza indirgemeye yardımcı olacak şekilde, web sitelerine özel kurallar oluşturulabilir. Önceden tanımlanmış olan yanlış-kullanım kurallarını kullanarak alarmlar üretir. Yeni kurallar tanımlamak için ayrı bir dile sahiptir. Snort, ağ paketlerini yakalamak için ikilik düzendeki tcpdump-formatlı dosyaları veya sade text dosyalarını kullanır.

Her kurala tetiklendiklerinde yapacakları hareket önceden atanır. Aşağıda Snort uyarı dosya verisi yer almaktadır:

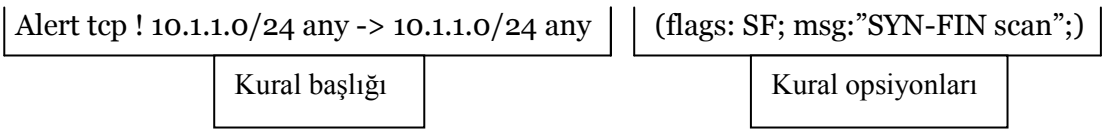
```
[**] NMAP TCP ping [**]
03/21-13:33.51:880120 1.2.3.4:1029 -> 192.168.5.5:80
TCP TTL:46 TOS:0x0 ID:19678
******A* Seq: 0xE4F00003 Ack: 0x0 Win: 0xC00
```

Bu uyarıya, “ACK” bayrağı olan fakat “ack” değeri 0 olan TCP paketlerini inceleyen bir kural sebep olmuştur. Bu durum, genelde, aktif konakçıları tespit etmeye çalışan bir “nmap” işaretidir [14].

Snort kural yapısı

Kural, iki parçadan oluşmaktadır. İlk bölüm, trafiğin dikkate alınması için içerisinde kimlerin olması gerektiğinin tarif edildiği kural başlığıdır. İkinci bölüm, dikkate alınacak trafiğin içinde nelerin olması gerektiğinin tarif edildiği kural opsiyonlarıdır. Bu bölümde paket başlığı bilgileri veya paketin veri kısmının içeriği yer alabilir.

Şekil 2.2’deki kuralda, eğer, 10.1.1.x olmayan bir ağın herhangi bir portundan 10.1.1.x olan ağın herhangi bir portuna giden TCP trafiğinde SYN ve FIN bayraklarının ikisi birden olağandışı bir şekilde yer alıyor ise "SYN-FIN scan" mesajıyla beraber bir uyarı verileceği ifade edilmektedir [14].



Şekil 2.2 : SNORT kuralı [14].

Örnek kural:

```
alert udp any any -> 192.168.5.0/24 31337 (msg:"Back Orifice";)
```

Örnek çıktı:

```
[**] Back Orifice [**]
04/24-08:49:21.318567 192.168.143.15:60256 -> 192.168.5.16:31337
UDP TTL:41 TOS:0x0 ID:49951 Len: 8
```

Yukardaki Snort kuralında,herhangi bir IP adresinden veya portundan kaynaklanan bir UDP paketi 192.168.5 alt-ağında 31337 numaralı port'a giderse alarm verilmesini, kaydedilmesini ve "Back Orifice" içerikli mesaj verilmesini istemektedir.

Aşağıdaki örnekte bir paketin veri kısmının içeriğinde yapılan arama gösterilmektedir.

Örnek kural:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 \
(msg: "EXPLOIT BIND tsig Overflow Attempt"; \
content: "|00 FA 00 FF|"; content: "/bin/sh";);
```

Örnek çıktı:

```
02/22-15:33:19.472301 ATTACKER:1024 -> VICTIM:53
UDP TTL:64 TOS:0x0 ID:6755 IpLen:20 DgmLen:538
Len: 518
<çıktıyı kısaltmak için önceki satırlar yazılmamıştır>
00 3F 90 E8 72 FF FF FF 2F 62 69 6E 2F 73 68 00 .?..r.../bin/sh.
0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D ................
1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D .. !"#$%&'()*+,-
2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C EB ./0123456789:;<.
07 C0 00 00 00 00 00 3F 00 01 02 03 04 05 06 07 .......?........
08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 ................
18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 ........ !"#$%&'
```

```
28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 ()*+,-./01234567
38 39 3A 3B 3C EB 07 C0 00 00 00 00 00 3F 00 01 89:;<........?..
02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 ................
D8 FA FF BF D8 F7 FF BF D0 7C 0D 08 04 F7 10 40 .........|.....@
22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 "#$%&'()*+,-./01
32 33 34 35 36 37 38 39 3A 3B 3C EB 07 C0 00 00 23456789:;<.....
00 00 00 3F 00 01 02 03 04 05 06 07 08 09 0A 0B ...?............
0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B ................
1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B .... !"#$%&'()*+
2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B ,-./0123456789:;
3C EB 07 C0 00 00 00 00 00 00 00 FA 00 FF <.............
```

Bu çıktıda, yük bölümündeki onaltılık karakterler sol tarafta, bu karakterlerin ASCII yorumlaması da sağ tarafta yer almaktadır. Yaratılan kuralda, güvenilir ağ dışından güvenilir ağdaki 53 numaralı hedef porta olan UDP trafiği aranmaktadır. Özellikle, birincisi 00 FA 00 FF ondalık ifadesi, ikincisi /bin/sh metin ifadesi olan iki string değerin varlığı aranmaktadır. Bu string değerler, yük bölümünde herhangi bir sırada bulunabilirler.

"resp" opsiyonu, zararlı bir aktivite belirlendiğinde otomatik olarak cevap verilmesini sağlamaktadır. Bu aktif cevap, bağlantıyı kesmek olabilir. Bir kural içerisinde tanımlanabilecek çok sayıda aktif cevap ve "resp" opsiyonları bulunmaktadır. Gönderici taraftaki konakçı soket bağlantısına, alıcı taraftaki konakçı soket bağlantısına veya her ikisine birden "reset" gönderilerek TCP bağlantısı kopartılabilir. Eğer saldıran paket UDP ise, UDP veri akışını bozmak için değişik ICMP mesajlar gönderilebilir. "Ağ, konakçı veya port ulaşılamaz" ICMP mesajı veya bu üç mesajın farklı kombinasyonları gönderilebilir.

Örnek kural:

```
alert tcp any any -> $HOME_NET 21 \
(msg: "FTP password file retrieval"; \
flags: A+; resp: rst_all; content: "passwd";)
```

Örnek oturum:

```
[root@verbo hping2-beta53]# ftp sparky
Connected to sparky.
```

```
220 sparky FTP server (SunOS 5.7) ready.
Name (sparky:root): jsmith
331 Password required for jsmith.
Password:
230 User jsmith logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /etc
250 CWD command successful.
ftp> get passwd
local: passwd remote: passwd
200 PORT command successful.
421 Service not available, remote server has closed connection
```

Yukardaki kural, "passwd" adlı parola dosyasını referans veren bir ftp sunucusuna olan bir bağlantıya aktif cevap verilmesini istemektedir. Snort, bu girişimi bozmak için bağlantının her iki tarafını da sıfırlar, çünkü "rst_all" resp opsiyonu seçilmiştir. Ftp oturumunun son satırına bakılırsa, saldırıcının "get passwd" komutunu girmesinden hemen sonra, bağlantının kapatıldığı görülmektedir.

Sızma alarmlarının ilişkilendrildiği bir çalışmada [15], Snort tarafından oluşturulan saldırı uyarılarının kümeleme algoritmalarında karakterize edilmesi için 40 adet özellik oluşturulmuştur.

```
[**] [1:718:7] TELNET login incorrect [**]
04/05-09:38:26.936288 172.16.114.50:23 -> 172.16.114.168:10332
TCP TTL:64 TOS:0x10 ID:2014 IpLen:20 DgmLen:66 DF
**AP*** Seq:0x801D2FDC Ack:0x6EDB5C7F Win:0x7C00 TcpLen:20
```

Yukardaki Snort uyarısından da görülebileceği gibi, bu paketteki TCP başlığında sadece ACK ve PSH TCP bayrakları set edilmiştir (***AP*** ifadesinden anlaşılmaktadır). Bu bilgi "tcpFlag*" özellik grubunda kodlanmıştır. "tcpFlagAck" ve "tcpFlagPsh" 1 değerini alırken diğer bayraklar 0 değerini almıştır. Snort uyarısındaki "portSrc" ve "portDest" özellikleri TCP kaynak ve hedef portları göstermektedir ve sırasıyla 23 ve 10332 değerlerini almışlardır. Yukardaki Snort uyarısının özellik değerleri Çizelge 2.2'de yer almaktadır.

Çizelge 2.2 : Kodlanmış Telnet Uyarısı [14].

portSrc	23	icmpType	0
portDest	10332	tcpFlag1	0
ipIsIcmpProtocol	0	tcpFlag2	0
ipIsImgpProtocol	0	tcpFlagUrg	0
ipIsTcpProtocol	1	tcpFlagAck	1
ipIsUdpProtocol	0	tcpFlagPsh	1
ipLen	20	tcpFlagRst	0
ipDgmLen	66	tcpFlagSyn	0
İpId	2014	tcpFlagFin	0
ipTos	16	tcpLen	20
ipTtl	64	tcpWinNum	31744
ipOptLsrr	0	tcpUrgPtr	0
ipPacketDefrag	1	tcpOptMss	0
ipReserveBit	0	tcpOptNopCount	0
ipMiniFrag	0	tcpOptSackOk	0
ipFragOffset	0	tcpOptTcl	0
ipFragSize	0	tcpOptTs2	0
icmpCode	0	tcpOptWs	0
icmpId	0	tcpHeaderTrunc	0
icmpSeq	0	udpLen	0

2.1.2 Gezgin etmenler

Etmen, özerkliğe sahip olan özel bir nesne çeşididir. Gezgin etmen, açık ve dağıtık sistemlerde kendi kodunu ve verisini taşıyarak makineden makineye gezebilme kabiliyetine sahiptir. Çalışması esnasında, ağın başka bir yerindeki kaynakları kullanmak için, oraya taşınabilir.

Geleneksel istemci/sunucu modelinde, sunucu önceden tanımlanmış hizmet arayüzlerini ortaya koyar, istemciler sunucuya veri göndererek hizmet isteğinde bulunur. Aksine, gezgin-etmen temelli model, istemcilere etmenler üzerindeki işlemler konusunda kendi tercih ettikleri yöntemi tanımlamalarına izin verir. Etmenler, sunucular arasında gezinerek görevlerini özerk bir şekilde yerine getirirler. Çoğu araştırmacıya göre açık ve dağıtık ortamlar için gezgin etmenler artık vazgeçilmez hale gelmiştir [16].

Gezgin etmen, ağlar arasında dolaşma, makinelerle etkileşime girme, bilgi toplama ve işlerini bitirince dağıtıcısına dönme kabiliyetlerine sahiptir [17]. Gezgin etmenlerin kullanılma nedenleri aşağıda sıralanmıştır:

- Ağ yükünü azaltmak,
- Ağdaki gecikmeleri engellemek,

- Asenkron ve otonom olarak çalışmak,
- Ortam değişikliklerine adapte olmak,
- Doğası gereği sağlam olmak ve hatalara toleranslı olmak [16].

Gezgin etmenler, analiz için konakçılar arasında veri taşınması yerine sızma belirleme kodunun verinin bulunduğu yere taşınması anlamında da çok kullanışlıdır [18]. Dağıtıcılarından koptuklarında dahi görevlerini yürütebildikleri için, kontrol ünitesinin göçmesi, devam eden sızma belirleme görevlerini durdurmayacaktır. Bu durum, sistemi daha güvenilir hale getirmektedir.

Her bir etmen, sistemin sadece küçük bir parçasını gözlemler. Basit bir etmen, tek başına, sızma belirleme sistemini oluşturamaz; çünkü görüş alanı sistemin küçük bir dilimiyle sınırlıdır. Fakat, eğer bir sistemde çok sayıda etmen çalışır ve birbirleriyle yardımlaşırlarsa, güçlü bir SBS oluşturulabilir. Etmenler birbirlerinden bağımsız olduklarından, SBS'in tamamını yeniden inşa etmeye veya SBS faaliyetlerinin akışını dahi bozmaya gerek olmadan, dinamik olarak sisteme eklenebilir ve sistemden çıkartılabilirler. Böylece, yeni bir saldırı tipi belirlenirse, yeni etmenler geliştirilebilir, sisteme eklenebilir ve özel bir güvenlik politikasına göre ayarlanabilirler [19].

Gezgin etmen teknolojisinin bu kadar yaygınlaşmasında, Java'nın popülaritesinin büyük katkısı olmuştur. Web sayesinde, gezgin etmen uygulamalarına geniş olanaklar sağlanmıştır. Java sanal makinesi ve onun dinamik sınıflarıyla beraber "serialization, remote method invocation, reflection" özellikleri gezgin etmen sınıflarının oluşturulmasını kolaylaştırmıştır. Aglet [20], Ajanta [21], Concordia [22], D'Agent [23], Voyager [24], Naplet [16], JINI [25], Gypsy [26], J-Seal2 [27], SHOMAR [28] and Jade [29] gibi çok sayıda gezgin etmen sistemi geliştirilmiştir.

Bu tez çalışmasındaki uygulamalarda, JADE (Java Agent DEvelopment Environment) kullanılmıştır. JADE, etmen modelini temel alan ve hem kablolu hem de kablosuz ortamlarda çalışabilen eşdüzeyler-arası (peer-to-peer) uygulamaların geliştirilmesi ve çalıştırılmasına imkân veren bir ara-yazılımdır. Etmenler, konaklamaları ve çalışmaları için gereken tüm hizmetleri sunan dağıtık kapsayıcılarda (container) bulunurlar [30].

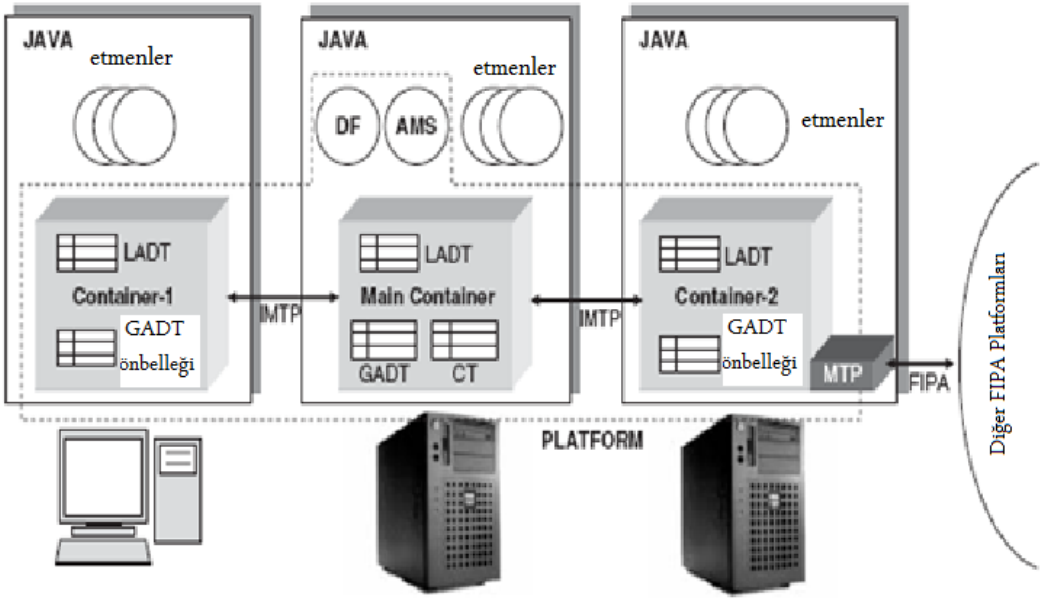
2.1.2.1 JADE

Eşdüzeyler arası iletişim, kablosuz veya kablolu ortam farketmeksizin, her bir eşdüzeyin başlatıcı veya cevaplayıcı rollerini oynayabileceği şekilde tamamen simetriktir. "Peer-to-peer" modelde, roller arasında bir ayrım yoktur ve her bir eşdüzey, inisiyatif ve kabiliyetlerin tamamına sahiptir. Her bir düğüm, iletişimi başlatabilir, bir isteğin öznesi veya nesnesi olabilir, pro-aktif olabilir. Uygulama mantığı, sunucu üzerinde yoğunlaşmaz, ağdaki bütün eşdüzeylere dağıtılır; her bir düğüm birbirini arayıp bulabilir, ağa herhangi bir zamanda herhangi bir yerden girebilir, bağlanabilir veya kopabilir [29].

JADE, programcılara aşağıdaki özellikleri sunar:

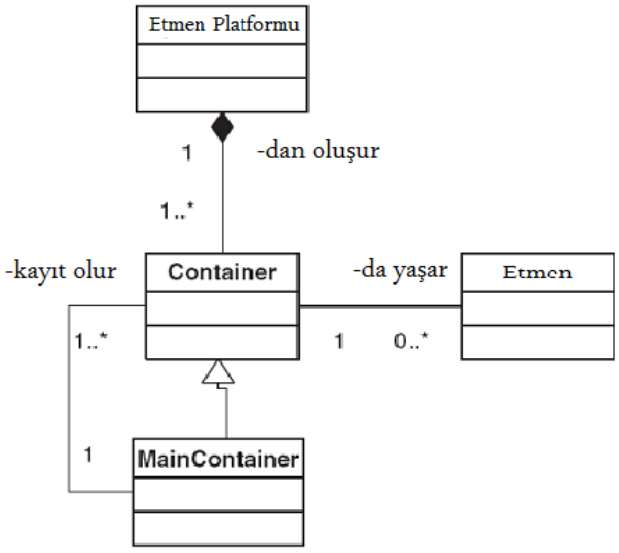
- Her biri aynı veya farklı makinelerde, ayrı birer iplik (thread) olarak çalışan, birbirleriyle kolayca iletişim kurabilen etmenlerin yer aldığı tamamen dağıtık bir sistem,
- FIPA (Foundation For Intelligent Physical Agents) ölçütlerine tam uyum,
- Asenkron mesajların etkin bir şekilde yerlerine ulaştırılması,
- Beyaz ve Sarı Sayfa uygulamaları,
- Etmen yaşam-döngüsü yönetimi: Etmenler yaratıldığında kendilerine, platformlarının beyaz sayfa servisine kayıt olmalarında kullanacakları, global olarak benzersiz olan bir tanıtıcı ve ulaşım adresi atanır. "Create, suspend, resume, freeze, migrate, clone ve kill" gibi basit API ve grafik araçlar sayesinde etmenlerin yaşamları yönlendirilir.
- Etmenlere taşınabilirlik desteği,
- Etmenler için abonelik mekanizması sayesinde ilgilenilen alandaki değişikliklerden haberdar olma imkanı,
- Programcıları hata ayıklama ve izleme konularında destekleyen grafik araçlar,
- Etkileşim protokolleri kütüphanesi,
- JSP, servlet, applet gibi web-tabanlı teknolojilerle bütünleşme,
- J2ME platformu ve kablosuz ortam desteği.

JADE platformu, Şekil 2.3'de görüldüğü gibi ağda dağıtık olarak yer alan etmen kapsayıcılarından oluşur. Etmenler, bu kapsayıcılarda yaşarlar. "Main container", platformda ilk olarak başlatılan ve diğer kapsayıcıların kendisine kayıt olmak yöntemiyle bağlanmak zorunda oldukları özel bir kapsayıcıdır. Diğer kapsayıcılar, bağlanma sıralarına gore "Container-1", "Container-2" gibi adlar alırlar.



Şekil 2.3 : JADE yapısı [30].

Şekil 2.4'deki UML diyagramında JADE'in temel yapısal elemanları arasındaki ilişkiler şema ile gösterilmiştir.



Şekil 2.4 : Temel elemanlar arasındaki ilişkiler [30].

“Main container”ın bazı özel sorumlulukları vardır:

- Platformu oluşturan kapsayıcıların tamamının ulaşım adreslerinin ve nesne referanslarının kayıt edildiği “Kapsayıcı Tablosunun (Container Table-CT)” yönetilmesi,
- Platformda yer alan tüm etmenlerin, halihazırdaki durumları ve yerleri bilgisi dahil, diğer bilgilerinin kayıt edildiği “Global Etmen Tanımlayıcı Tablosunun (Global Agent Descriptor Table-GADT)” yönetilmesi,
- Beyaz sayfa ve sarı sayfa hizmetlerini veren Etmen Yönetim Sistemi (The Agent Management System-AMS) ve Adres Rehberi Kolaylaştırıcısı (The Directory Facilitator-DF) etmenlerine konakçılık yapılması.

Her kapsayıcının yerel olarak yönettiği GADT önbelleği olduğu için “Main container”ın sistem darboğazı olması sözkonusu değildir. Kapsayıcı, kendisine gönderilen bir mesajın alıcısının yerini tesbit etmek için öncelikle LADT ve GADT önbelleğinde arama yapar. Eğer arama başarısız olursa “main container” ile irtibata geçilir ve sonuç kendi önbelleğine de kaydedilir [30].

Etmen, bir hedefle sadece mesaj göndererek haberleşir. Etmenler bir isimle nitelendirilirler (mesaj göndermek için hedef nesnenin referansına ihtiyaç yoktur), bunun sonucu olarak, haberleşen etmenler arasında geçici bağımlılık yoktur. Gönderici ve alıcı aynı zamanda kullanılabilir durumda olamayabilir. Alıcı hiç olmayabilir (henüz olmayabilir) veya gönderici tarafından doğrudan bilinmeyebilir; bu durumda gönderici hedef olarak bir özelliği (“futbolla ilgilenen tüm etmenler”) belirtebilir.

JADE, kodun ve yürütme durumunun gezginliğini desteklemektedir. Etmen, bir konakçıdaki koşturmasını durdurabilir, farklı bir uzak konakçıya taşınabilir (o konakçıda etmen kodunun önceden yüklenmiş olmasına gerek yoktur) ve durdurulduğu noktadan yürütmesine yeniden başlar. Bu işlevsellik sayesinde, çalışma-zamanında uygulamayı etkilemeyecek şekilde gezgin, etmenleri daha az yüklü olan makinelere taşıyarak hesaplama yükü dağıtılabilir. Platform, aynı zamanda birden fazla konakçı üzerine dağıtılabilen bir isimlendirme hizmetine (her bir etmenin benzersiz bir isminin olması sağlanır) ve sarı sayfalar hizmetine sahiptir [29].

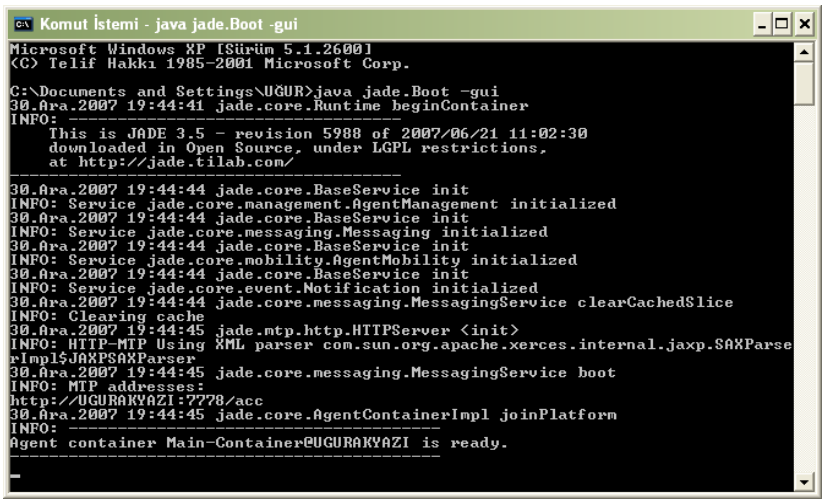
Platform'un Başlatılması

Bir etmenin platforma yüklenebilmesi için sınıf dosyasının CLASSPATH aracılığıyla ulaşılabiliyor olması gerekir.

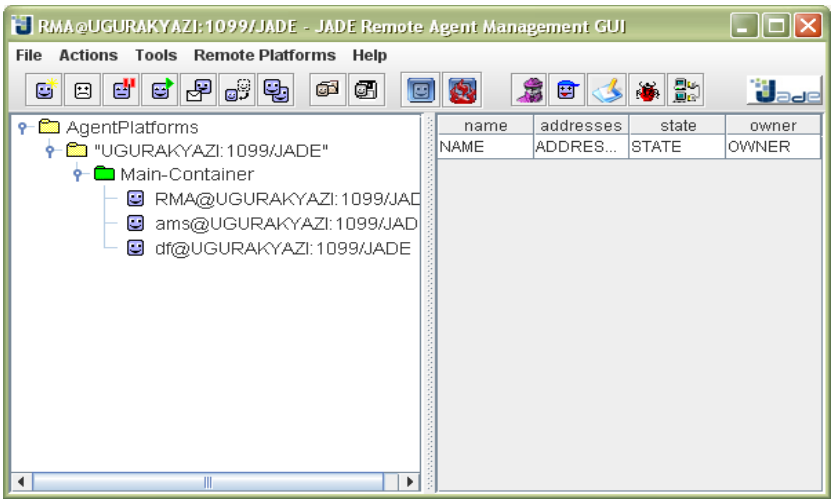
prompt> Java jade.Boot –gui

komutuyla JADE Şekil 2.5'de görüldüğü gibi çalıştırılır. "–gui" seçeneği Remote Monitoring Agent (RMA) tarafından sağlanan Şekil 2.6'daki JADE grafik arayüzünü başlatır. "Main container" başlatıldıktan sonra, başka konakçılarda aynı platforma ait başka kapsayıcılar da başlatılabilir. Mesela, "Main container"ın başlatıldığı makinenin adı UĞURAKYAZI ise aşağıdaki komut bir kapsayıcı başlatacak ve –host ile belirtilen konakçıda başlatılmış olan "Main container"a ekleyecektir.

prompt> java jade.Boot -container -host UĞURAKYAZI



Şekil 2.5 : jade.Boot komutu.



Şekil 2.6 : JADE grafik arayüzü.

2.1.3 Dağıtık saldırılar

2.1.3.1 TCP/IP İletişim Protokolü

Bu bölümde, internette en sık kullanılan protokollerin temellerini anlamak için gerekli olan bazı bilgiler verilecektir. Ağa yapılan saldırılar, genelde protokollerdeki açıkları kullandıkları için ve bu saldırıların nasıl çalıştıklarını daha iyi anlamak gerektiği için, bu bilgiler önem taşımaktadır.

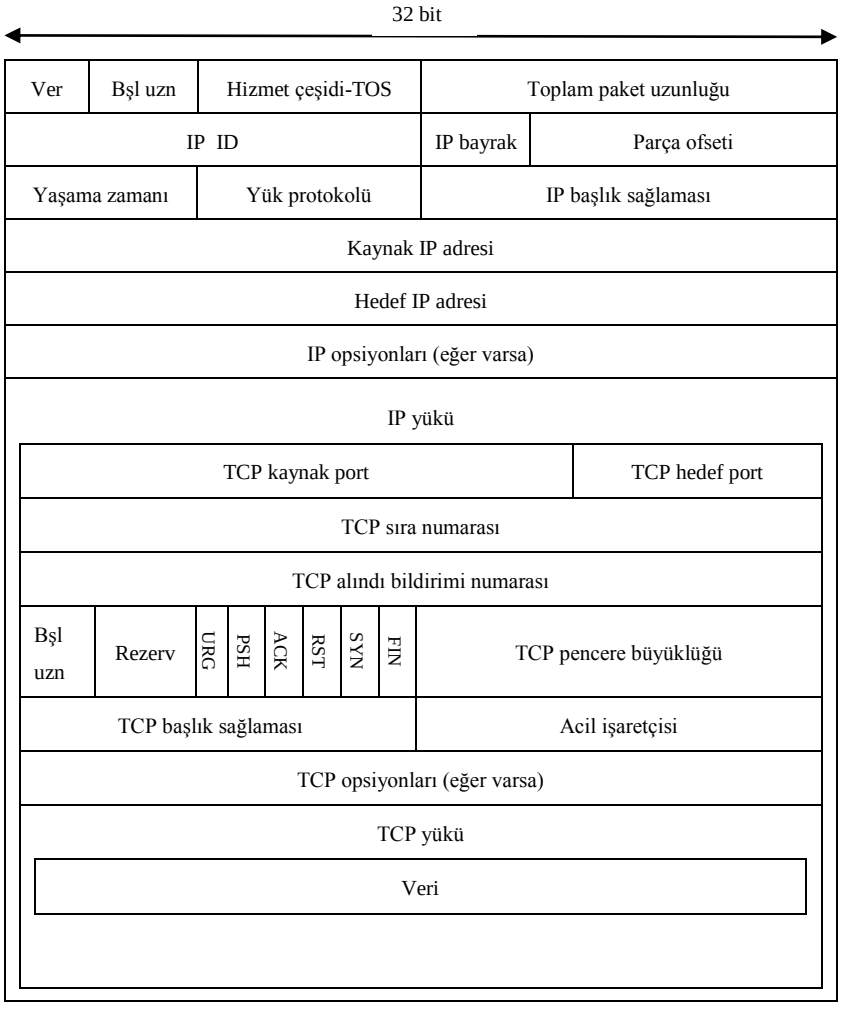
TCP ve IP, bir açık sistem bağlantısı için OSI modelin alt-basamak protokollerindendir. Kabaca, TCP ulaşım katmanını ve IP ağ katmanını temsil ederler. Şekil 2.7'de TCP ve IP protokollerini kullanarak iletişim kuran bir ağ paketinin protokol başlığı görüntüsü yer almaktadır. TCP, OSI katmanı olarak IP'den daha üstte olduğu için, şekilde görüldüğü gibi, TCP paketleri IP paketlerinin içinde yer alır. IP'nin iki adet öncelikli işlevi vardır:

- Ağdaki iki uç-nokta (IP adresleri) arasında veri bloklarının bağlantısız olarak aktarılmasını sağlamak,
- Farklı paket büyüklüğü kabiliyetindeki ağlara uydurmak için, veri bloklarının parçalanması ve tekrar birleştirilmesini sağlamak.

IP protokolünün nasıl çalıştığını anlamak için Şekil 2.7'de gösterilen IP başlık alanlarını bilmek gerekmektedir:

- *Versiyon:* Kullanılan IP protokolünün versiyonunu gösterir.
- *Başlık uzunluğu:* IP başlığının 32-bitlik kelime olarak uzunluğunu ifade eder.
- *Hizmet çeşidi (Type of service-TOS):* Paketin alması gereken hizmet kalitesini gösterir. Bu alanın alacağı değer, aslında bir üst-katman protokol tarafından verilir ve canlı internet telefon trafiği gibi düşük seviyede gecikme gerektiren trafiğin ayırt edilmesinde kullanılabilir.
- *Toplam paket uzunluğu:* Bütün paketin uzunluğunu byte cinsinden verir.
- *IP paket ID'si:* Paket için bir belirleyici numarası atar. Bu alan, başka bir ağda parçalanmış olan IP paketlerini tekrar birleştirmek için kullanılabilir.

- *IP bayrakları:* Parçalanmayla ilgili olarak iki paket bayrağını gösterir. Bayrağın birisi paketin parçalanıp parçalanmadığını gösterirken, diğeri de bu paketin aynı büyük paketin parçalarının sonuncusu olup olmadığını göstermektedir.
- *Parça ofseti:* Paketin parçalar serisindeki yerini gösterir.
- *Yaşama zamanı (Time to live-TTL):* Paketin ağdan atılmadan önce atlayabileceği düğüm sayısı için bir sayaç tutar, böylece paketin yolunda yanlış ayarlanmış bir yönlendirici varsa paketin sonsuz olarak iletilmesi engellenmiş olur. Her bir düğüm, bu değeri bir azaltır ve değer sıfır olunca paket atılır.



Şekil 2.7 : TCP/IP paket başlığı [15].

- *Yük prokolü:* Paketin yük bölümünde hangi protokolün taşınacağını belirtir. Bu durumda TCP olacaktır.

- *IP başlık sağlaması:* IP başlığının bütünlüğünü kontrol etmeye yardımcı bir değer verir.
- *Kaynak/hedef IP adresi:* Paketi gönderen kaynak düğüm ve paketin gitmesi istenilen hedef düğüm belirtilir. IP adesleri herbir elemanı [0-255] aralığında değer alan "a.b.c.d" ifadesi ile gösterilir.
- *IP opsiyonları:* Eğer varsa, güvenlik gibi IP paket opsiyonlarını belirtir.
- *IP yükü:* Bir üst-katman protokol için (bu durumda TCP) veri barındırır.

Uçtan-uca güvenirlik, akış kontrolü ve paket sıralama görevleri IP standartları dışında kalır ve TCP tarafından gerçekleştirilir. Bu özellikler TCP'yi güvenilir bir protokol yapar. TCP başlık alanlarından olan TCP bildirim numarasının ve TCP sıra numarasının özel önemi vardır. Düğümler bu alanları kullanarak bir TCP bağlantısı kurarlar ve diğer düğümün hangi veriyi aldığını senkronize ederler. Bu numaralar bağlantı kurma aşamasında başlatılırlar ve bağlantı esnasında düğümler arasında geçen verinin boyutunu yansıtması için güncellenirler. Bir TCP bağlantısı kurmak için, iki taraf üçlü-el sıkışma gerçekleştirirler. A'nın B ile oturum açmak istemesi durumunda [15]:

1. A, TCP SYN bayrağı set edilmiş olarak (bir SYN paketi) ve B'ye özel bir başlangıç sıra numarası olan TCP/IP paketi gönderir.
2. B, TCP SYN ve ACK bayrakları set edilmiş (bir SYN/ACK paketi) ve A'ya özel bir başlangıç sıra numarası olan bir TCP/IP paket cevabı göndererek A'nın SYN paketine bildirimde bulunur.
3. A, B'nin SYN/ACK paketine cevap olarak ve el sıkışma işlemini bitirmek için TCP ACK bayrağı set edilmiş bir paket gönderir.

Diğer TCP alanları aşağıda açıklanmıştır:

- *Kaynak/hedef TCP port:* IP paketin kaynak ve hedef düğümlerindeki TCP port numaraları belirtilir. Düğüm çiftleri arasında eşzamanlı olarak çok sayıda TCP bağlantısı kurulabilmesi için bu numaralar gerekmektedir. Paketteki TCP port, bazen hangi hizmetin istenildiğini belirtmek için faydalı olabilmektedir (port 80'nin HTTP'yi belirtmesi gibi).
- *Başlık uzunluğu:* TCP başlığının 32-bitlik kelime cinsinden uzunluğunu belirtir.

- *Rezerv:* Gelecekte kullanmak için rezerve edilmiştir.
- *TCP bayrakları:* Bağlantının durumunu kontrol eder. Birden fazla bayrak set edilebilir, ama bazı bayrak kombinasyonları yasal değildir.
    - *URG:* "URGENT" bayrağı, acil verileri ifade eder.
    - *PSH:* "PUSH" bayrağı, pakette veri gönderildiğini ifade eder.
    - *ACK:* "ACKNOWLEDGE" bayrağı, önceki pakette veri alındığını veya bir el sıkışma işleminin yürümekte olduğunu gösterir.
    - *RST:* "RESET" bayrağı, birşeylerin yanlış olduğunu ve bağlantının kaybedildiğini gösterir. İstenilen portta bir hizmet verilmediğini belirtmek için de kullanılır.
    - *SYN:* "SYNCHRONIZE" bayrağı, üçlü-el sıkışma gerçekleştirme isteğini gösterir.
    - *FIN:* "FINISH" bayrağı, TCP bağlantısının bitirilmek istendiğini gösterir.
- *TCP pencere büyüklüğü:* Bağlantıdaki akış kontrolü için alıcının gelen-veri tampon büyüklüğünü gösterir.
- *TCP başlık sağlaması:* TCP başlığının bütünlüğünü doğrulamaya yardımcı olur.
- *Acil işaretçisi:* Yükteki acil verinin ilk byte'ına olan offset değerini gösterir.
- *TCP opsiyonları:* Eğer varsa, TCP opsiyonlarını belirtir.
- *TCP yükü:* TCP'nin taşıdığı yükü barındırır. Bir HTTP hareketinin parçası gibi üst-katman protokolü bilgisidir.

2.1.3.2 Dağıtık saldırı çeşitleri

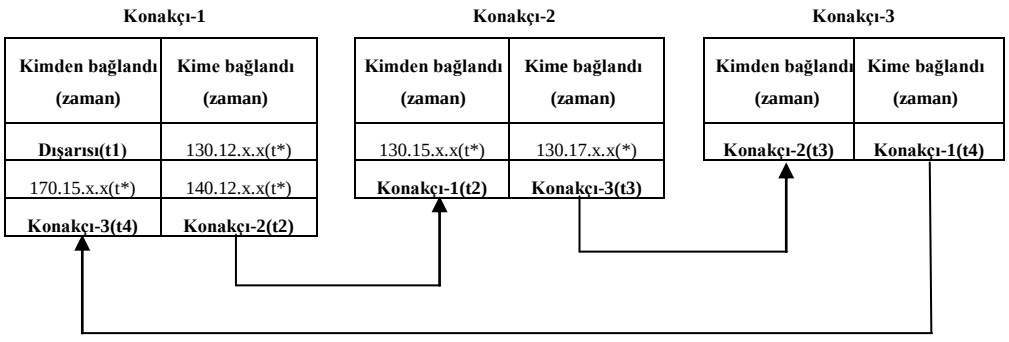
1. Kapı topuzunu-tangırdatma saldırısı (Doorknob-rattling attack): Bu saldırıda, saldırıcının amacı, yaygın kullanılan hesap ve şifre birleşimlerini kullanarak bir sistemdeki koruması zayıf olan konakçıları tespit etmek ve içlerine sızmaktır. Saldırıcı, her makinede farklı hesaplar kullanarak, başarılı ve başarısız, az sayıda oturum açma denemesi yapar. Bu denemeler, tek bir konakçıya kurulu, yüksek "belirleme eşikleri" olan SBS'ler tarafından

belirlenemez. Bütün konakçılardaki başarısız oturum açmalara ilişkin veriler toplanıp birleştirilmesi gerekmektedir.

Örneğin, iki adet başarısız giriş denemesi normal olarak değerlendirilebilir, ama 30 tane başarısız deneme bir şüphe doğuracaktır. Eğer saldırıcı, 15 makinenin her birine ikişer kez "doorknob rattle" denemesi yaparsa, bu 15 makinenin hiç birisi, bilgi ilişkilendirmesi yapılmadan bu davranışı ciddiye almayacaktır. Halbuki bilgi ilişkilendirilmesi yapıldığında, bu başarısız girişlerin aslında bir saldırının parçaları olduğu anlaşılacaktır [31].

2. Zincir/Döngü saldırısı (Chain/Loop attack): Bu saldırıda, sızıcı, başlangıç noktasını gizlemek için değişik konakçılar arasında dolaşarak bağlantılar zinciri oluşturur. Döngü saldırısı durumunda, bağlantılar zinciri, bağlantının kaynağının izinin takip edilmesini zorlaştıracak şekilde bir döngü içerisindedir. Zincir saldırılarının çoğu ağ içerisinden yapılmaktadır.

   Şekil 2.8'deki bağlantıların kaynağını belirlemek için, konakçılardan, zaman bilgileri olan giriş ve çıkış verileri toplanmış ve ilişkilendirilmiştir. Koyu renkli verilerdeki t1>t2>t3>t4 zaman sıralaması, Konakçı-1'in Konakçı-2'ye, Konakçı-2'nin Konakçı-3'e ve Konakçı-3'ün Konakçı-1'e bağlı olduğu bir döngü saldırısını göstermektedir. t(*), bir zincir/döngü saldırısıyla sonuçlanmayan bağlantıların zamanını belirtmektedir [3].



Şekil 2.8 : Zincir/döngü saldırısı örneği [3].

3. Dağıtık port tarama saldırısı (Distributed port scanning attack): Port tarama faaliyeti, belirlenmesini zorlaştırmak için birden fazla konakçı üzerine dağıtılabilir. Dağıtık port tarama, ancak değişik konakçılardaki tarama faaliyetleri birleştirilir ve analiz edilirse belirlenebilir.

4. Dağıtık Hizmetin Engellenmesi saldırısı – DDoS: Bu saldırıda, geniş bir alana yayılmış olan çok sayıdaki konakçı kullanılarak, bir sisteme karşı DoS saldırıları başlatılır. TFN, Trinoo, and Stacheldraht, örnek DDoS saldırı araçlarıdır. Bütün DDoS araçları, DoS saldırısı başlatmak için kullanılan konakçılara hayalet program (daemon) yüklerler. Bir ana program, hayalet program gruplarını kontrol eder; saldırı yapan sistemde yüklü olan bir istemci program da ana programı kontrol eder. Hayalet programlar, hedef sistem üzerine SYN taşkını (flooding), UDP taşkını, ve ICMP taşkını saldırıları gibi DoS saldırıları başlatabilirler [3].

2.1.3.3 Dağıtık hizmetin engellenmesi saldırısı

DDoS saldırıları, verilen bir IP adresindeki bilgisayarın sağladığı hizmetleri engellemek veya zayıflatmak amacıyla tasarlanmıştır. Bu saldırının gerisinde yatan asıl nedenler aşağıda anlatılmaktadır:

1. İntikam: İnternet "chat"i sırasındaki önemsiz tartışmaların bir parçası olarak, bir rakibin uzaktan bağlantısını veya ev bağlantısını koparmak için yapılabilir.
2. Kanıtlama: DDoS saldırıları normalde, bir kişinin kontrolü altında, aynı virus bulaştırılan bilgisayar toplulukları olan, "botnet"leri kullanırlar. Korsan aleminde kiralanmadan veya satılmadan önce "botnet"in büyüklüğü ve gücünü ispat etmek için, DDoS saldırıları kullanılabilmektedir. Bu amaçsız saldırıların çoğunda rastgele bir kurban seçilir.
3. Gasp: Rus organize suç örgütlerinin favorisi olan bu yöntemde, kurban olarak seçilen e-ticaret ve yasal çevrim-içi kumar sitelerinin işlerine devam edebilmeleri için onbinlerce dolar fidye istemek amacıyla DDoS saldırıları kullanılmaktadır. Hapiste yatan faillerle yapılan röportajlarda, kendi isteklerini kale almayan kurbanlarından vazgeçtiklerini ve yeni anlaşmalar yapabilmek için başka hedeflere saldırdıklarını itiraf etmişlerdir.
4. Rekabet avantajı: İş kaybı veya utanç yaşaması ve mevcut veya potansiyel müşterilerini "çok sık olarak saldırılardan dolayı hizmet veremeyen bir rakip" ile çalışmaya zorlamak için, bir rakibin web sitesini göçertmek amacıyla DDoS hizmetleri kiralanabilmektedir.

5. Yandakilere verilen zarar: Aynı sunucu ve IP adresinde binlerce web sitesinin sunulabilmektedir. Sitenin bir tanesine yapılan bir saldırı, sitelerin tamamını çalışmaz hale getirebilmektedir. Internet topolojisinden dolayı, büyük saldırılar, saldırı henüz istenilen hedefe ulaşmadan önce, bağlantıyı sağlayan firmaları sakat bırakmaktadır. Hiç bir nedeni yokken, milyonlarca kullanıcının bağlantısını sekteye uğratacak şekilde, yönlendiriciler de bu tür saldırılara maruz kalabilmektedirler.
6. Birleşik saldırılar: Gerçek dünyadaki geleneksel saldırılara (banka hırsızlığı, terorist bombalaması) katılarak, paniğe sebep olmak için iletişim hatlarını bozmak ve ilk karşı koyucu güçleri engellemek için kullanılabilir.
7. Politik saldırılar: Bu saldırılar, sonradan politik amaçla kullanılacak olan, devlet içerisinde özel bir fonksiyonda kullanılan yerlerdeki IP adreslerini etkiler. Protesto saldırıları da genelde bu grup içerisinde değerlendirilir [32].

DDoS saldırıları, internet hizmetlerini çok ciddi zararlara uğratabilmektedir. İlk DDoS saldırısı, 1999 yılı Ağustos ayında gerçekleştirilmiştir. 227 farklı sisteme Trinoo adlı DDoS aracı yerleştirilerek Minnesota Üniversitesi'ndeki bir bilgisayara "taşkın" şeklinde saldırı düzenlenmesi sonucunda, sistem iki gün boyunca çalışamaz durumda kalmıştır. Basında yer bulan ilk saldırı ise, 2000 yılı Şubat ayında gerçekleştirilmiştir. 7 Şubat'ta Yahoo web sayfası üç saat boyunca ulaşılamaz hale getirilmiş; ertesi gün de Amazon, Buy.com, CNN ve eBay web siteleri DDoS saldırısı yüzünden önemli oranda yavaşlatılmıştır. Tahminlere gore, Yahoo ulaşılamadığı üç saat boyunca $500,000 kadar, Amazon.com ise 10 saat boyunca $600,000 kadar e-ticaret ve reklam geliri kaybetmiştir [33]. Aylarca süren araştırmalar sonrasında, "mafiaboy" takma ismini kullanan 15 yaşında bir Kanadalı genç, bu saldırıların sorumlusu olarak tutuklanmıştır.

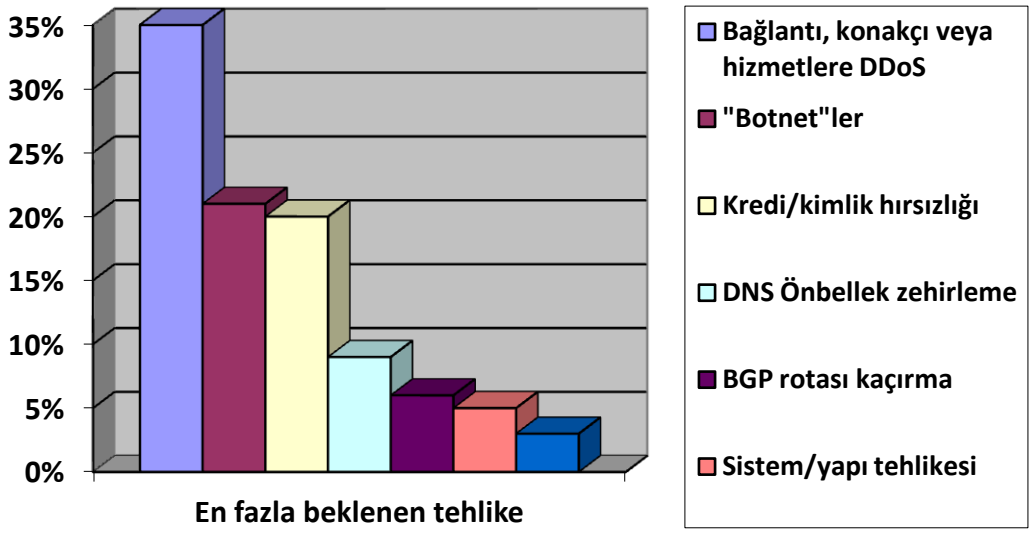
Enerji dağıtımı, iletişim, ulaşım, acil servisler gibi kritik yerlere yapılacak olan saldırılar, ekonomik zarar vermekle beraber can ve mal kayıplarına da sebep olacaktır. Örneğin, Ağustos 2003'de MSBlaster solucanı Maryland Motorlu Araçlar İdaresini bir günlüğüne kapattırmıştır. Aslında bilgisayarları MSBlaster'dan koruması beklenilen, "Welchi", "Welchia" veya "Nachi" diye adlandırılan bir başka solucan ise, Air Canada'nın bilet kotrol sistemini alt üst etmiş, ABD Deniz Kuvvetleri intranetindeki tasnif dışı bilgisayarlara sızmış ve ABD'nin üçüncü büyük

demiryolları işletmesi olan CSX şirketindeki tren sinyalizasyon sistemini kısa bir süreliğine kapatmıştır. Sinyaller çalışmadığı için tren gecikmelerine ve iptallerine neden olmuştur.

DDoS saldırıları, karşı görüşlü kişilerin yayınlarına sansür koymak amacıyla da kullanılabilmektedir. En son Irak savaşında, Katar haber kanalı olan "El-Cezire"nin ingilizce web sitesinin bu tür bir saldırıya maruz kalması, bu konuda yakın zamanda meydana gelen bir örnektir. ABD'deki 11 Eylül terör saldırılarından sonra, internetin de terör kurbanı olabileceği endişesi yayılmıştır. Bu tarihten sonraki DoS saldırılarında bir artış olduğu gözlemlenmiştir [4].

Diğer kullanıcıların makinelerine yönetici hesabıyla erişim imkânı sağlayan gelişmiş araçlar, internette ücretsiz olarak bulunabilmektedir. Tecrübesiz kullanıcılar bile bu araçları kolaylıkla kullanabilmektedir. Bir makineye giriş elde edildikten sonra, o makine saldırıcı tarafından kontrol edilen "master" bilgisayar kontrolündeki bir "zombi" haline dönüşmektedir.

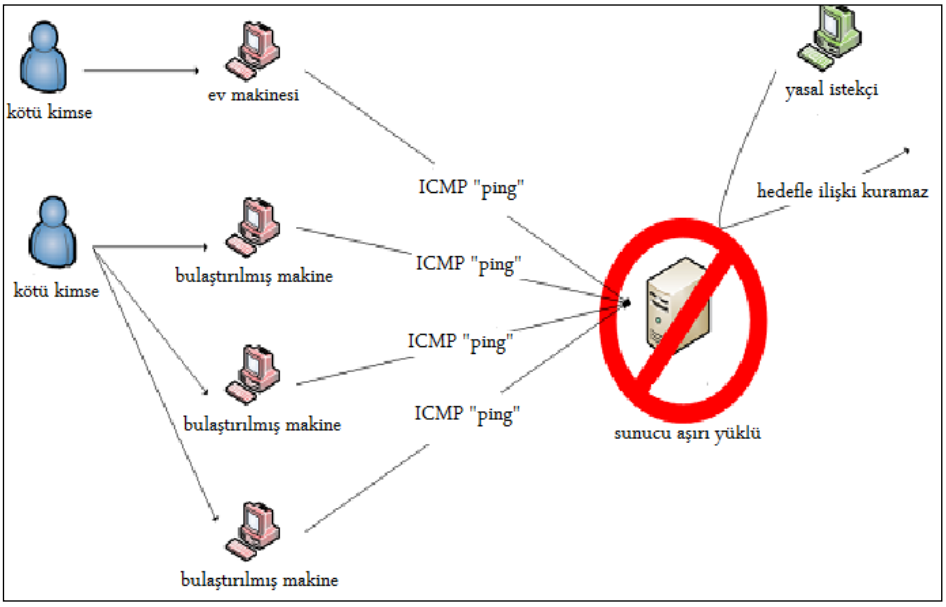
Arbor Network tarafından yayınlanan "Worldwide Infrastructure Security Report-2009"da [34] cevaplayıcılara, "gelecek 12 ayda hangi tehditlerin büyük işlemsel problemlere neden olacağına inandıkları" sorulmuştur. Şekil 2.9'da görüldüğü gibi "hizmetler, konakçılar veya bağlantılara olan DDoS" tehditi yaklaşık %35 ile ilk sıraya yerleşmiştir. Onu %21 ile "botnet"ler ve onlardan kaynaklanan aktiviteler takip etmiştir.



Şekil 2.9 : Beklenilen saldırıların çeşitliliği [34].

DDoS saldırı çeşitleri

En basit ve en eski yöntemlerden birisi olan "ICMP Ping taşkını" yönteminde, Şekil 2.10'da görüldüğü gibi kurallara uyan vatandaşlar, ev bilgisayarlarından basit bir şekilde "ping" ve bir IP adresi tuşlarlar. Bu şekildeki komutların binlercesinin birleşik etkisi, bir web sitesiyle olan iletişimi bozmak için yeterli olabilmektedir. Bu tür saldırıların çoğunda olduğu gibi, bir "botnet" içerisindeki çok sayıda makine üzerinde bu işlemi otomatik olarak yaptıracak programlar bulunmaktadır.

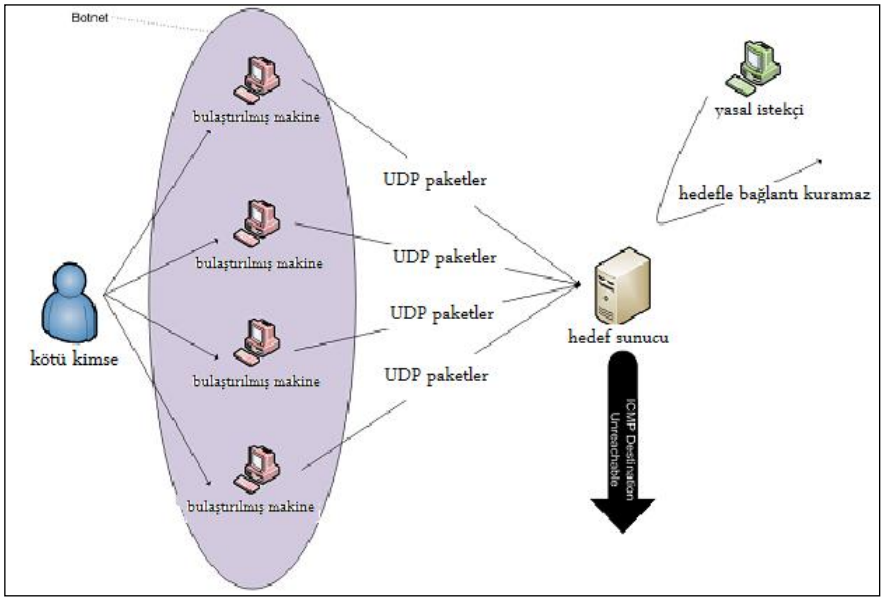


Şekil 2.10 : ICMP Ping taşkını.

Şekil 2.11'de gösterilen "UDP taşkını" saldırısında, hedef makinedeki değişik portlara UDP paket akımı gönderilir. Hedef makine bu paketlerin birini aldığında, programlarından herhangi birisinin ilgili porta gelen veriyi kabul etmek için kodlanıp kodlanmadığını kontrol etmek durumundadır. Büyük olasılıkla, o porta gelen veriyi dinleyecek olan bir program olmadığını farkeder ve normal olarak gelen paketin sahibine "bir problem var, senin için bu portta birşey yok" anlamında bir cevap mesajı gönderir. Bu cevap mesajına "ICMP Hedef Ulaşılamaz (Destination Unreachable)" paketi denilmektedir.

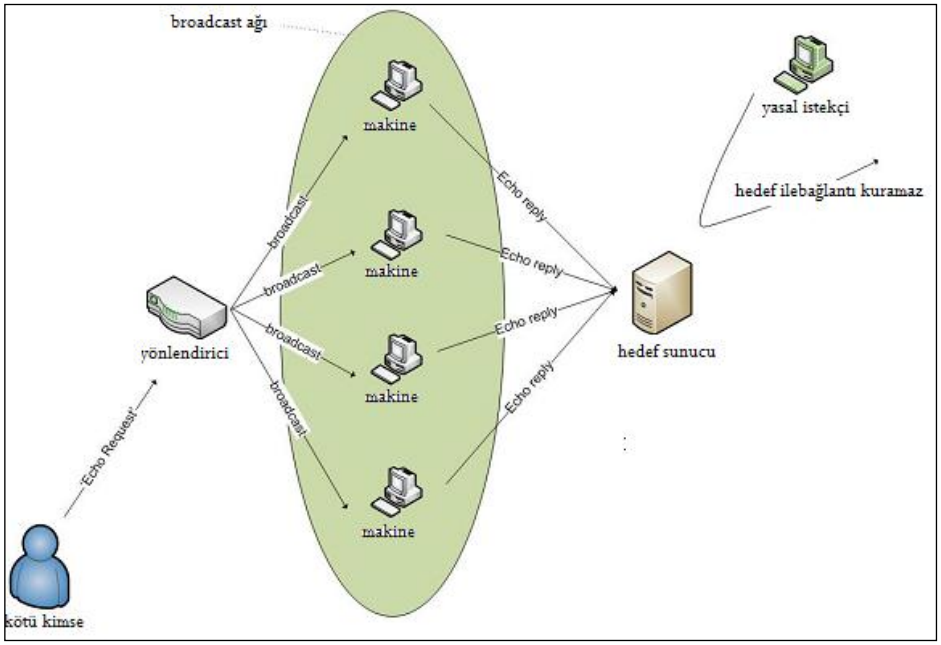
Bu UDP paketlerinden yeterince gönderilince, hedef makine "ICMP Destination Unreachable" paketleriyle cevap vermekle meşgul olacak ve yasal isteklerin hiçbirisi karşılanamayacaktır. UDP ile paketlerin nereden geldiklerini saklamak "spoofing"

yöntemiyle oldukça kolaydır. Böylece, saldırıcı makinelerin "ICMP Destination Unreachable" paketlere boğulması engellenmiş olur.



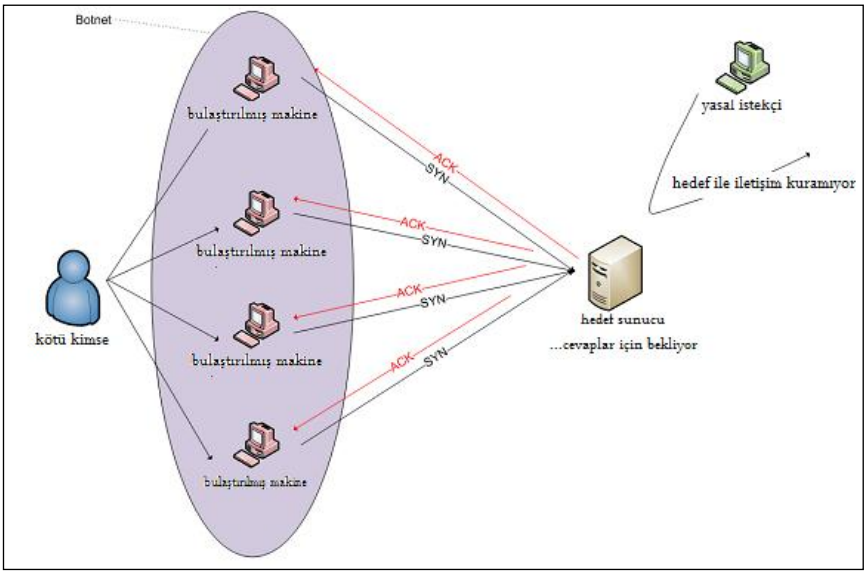
Şekil 2.11 : UDP taşkını.

"Smurf" saldırısında, bir ağdaki yönlendirici cihaza "Echo Request" paketi gönderilir, fakat gönderilen veri paketinin kaynak adresi saldırı hedefinin IP adresi ile değiştirilir. Bu "echo" isteği ağda "broadcast" adresi ile ulaşılabilen makinelerin hepsine gönderilir. Mesajı alan her bir makine, Şekil 2.12'de gösterildiği gibi, istenilen "echo" cevap mesajını hedef makineye gönderir.



Şekil 2.12 : Smurf saldırısı.

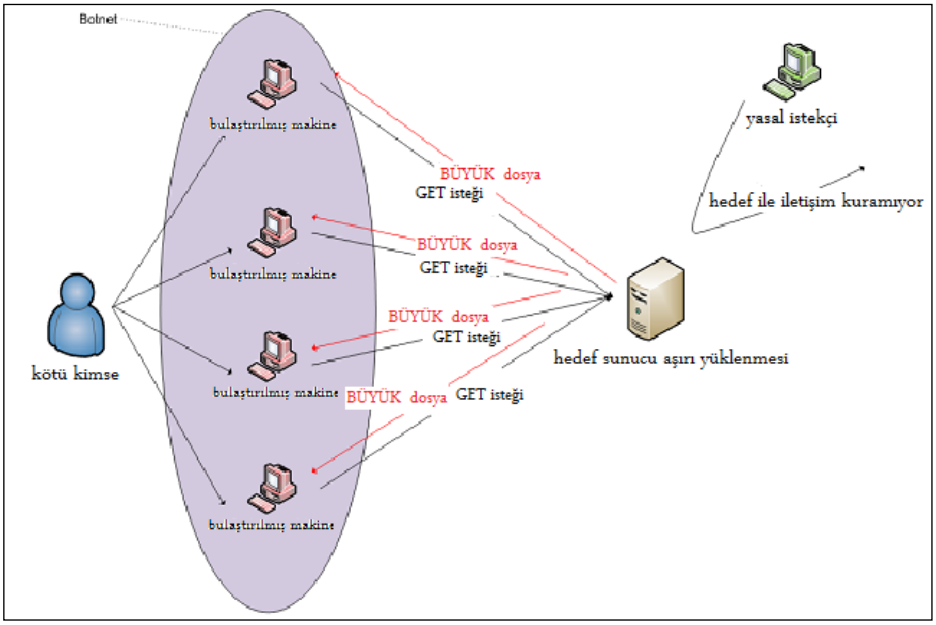
Bir web sitesine bağlanıldığında, iki bilgisayar, planladıkları iletişim yönteminde anlaşmak için kısa bir sohbete girerler. Bu anlaşmanın başlangıcında "SYN" (synchronization) denilen özel bir paket gönderilir; karşı taraftaki bilgisayar bir cevap gönderir ve karşılıklı konuşmanın devamını bekler. Şekil 2.13'de gösterilen "SYN taşkını" saldırısında, bu paketlerden sunucuya çok sayıda gönderilir, karşılığında yine çok sayıda "ACK" (acknowledgement) denilen cevaplar oluşur. Bu durumda, sunucudaki bütün kullanılabilir olan kaynaklar harcanır ve yasal olan diğer trafiğe cevap verilme imkanı kalmaz.



Şekil 2.13 : SYN taşkını.

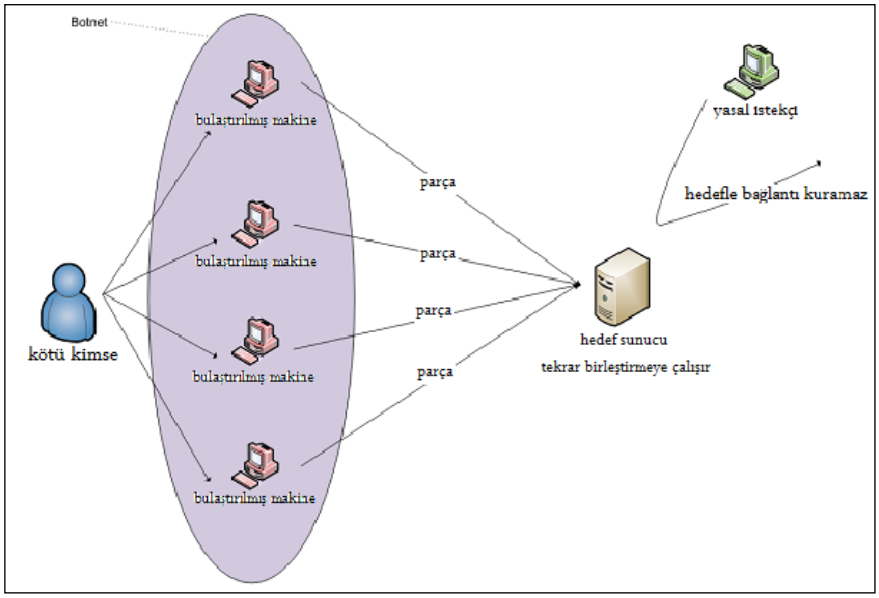
Bir web sitesi ziyaret edildiğinde, istemci bilgisayar görmek istediği sayfa için "GET" isteği ile istekte bulunur. Aynı yolla, kötü niyetli kimseler, Şekil 2.14'deki gibi bir "botnet"in parçası olan makineleri, büyük bir dosya isteğinde bulunmaları için ayarlayabilirler. Bu işlem düzenli olarak çok sayıda makine ile yapılırsa, o sitenin içeriğine olan yasal istekler sunucuya ulaşamaz, çünkü kullanılabilecek bütün bağlantılar çoktan doldurulmuştur.

İnternetteki normal işlemler sırasında, büyüklüklerinden veya geçtikleri ağlardaki bazı sınırlandırmalardan dolayı verilerin paketleri bölünmek veya parçalanmak (fragment) zorunda kalabilir. Yönlendiriciler, ateş duvarları ve sunucular gibi internet altyapısının değişik parçaları, tam orjinal paketle çalışmak veya analiz etmek için bu paket parçalarını tekrar birleştirmeye ayarlanmış olabilirler.



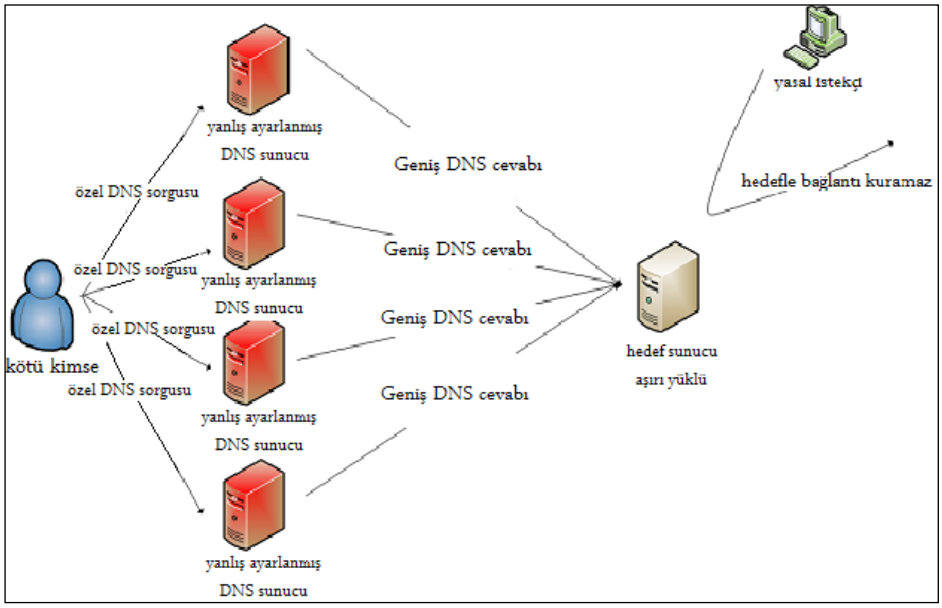
Şekil 2.14 : GET isteği.

Şekil 2.15'te yer alan "Frag taşkını" iki şekilde çalışır; birincisi, diğer DDoS saldırıları gibi, yönlendiricileri, ateş duvarlarını, sunucuları ve ağ bağlantılarını çok miktarda veri ile boğma yöntemidir. İkincisi, tekrar birleştirme ihtimali düşük olan özel hazırlanmış paket parçalar gönderilmesidir. Bu paketlerde, diğer parçalar hakkında bilgi taşıması gereken ilk parça yaratılmamış ve gönderilmemiştir. Bu durum, tekrar birleştirme mekanizmalarını boğacak ve makinelerin kilitlenmesine ve göçmesine neden olacaktır. Bunların yanında, ağ bağlantılarını da atılması gereken çöp paketlerle dolduracaktır.



Şekil 2.15 : "Frag" taşkını.

Şekil 2.16'da gösterilen DNS sunucular, internetin çalışması için kritik öneme sahiptirler; bilgisayarlara ziyaret etmek istedikleri web sitelerinin IP adreslerini verirler. DNS sunucular bazen, bir bilgisayarın bu tür sorgulamalar yapmasına ve cevabın gönderileceği yeri yanlış olarak vermelerine imkan tanıyacak şekilde, yanlış ayarlanmış olabilmektedirler. Çok sayıdaki cevaplar, saldırılmak istenilen hedef makineye yönlendirilebilir. Bu durumda, çoğu DDoS saldırısında olduğu gibi, ağdaki veri yolları çöple dolar ve yasal trafiğin geçmesine engel olur.



Şekil 2.16 : DNS çoğaltma (amplification) saldırısı.

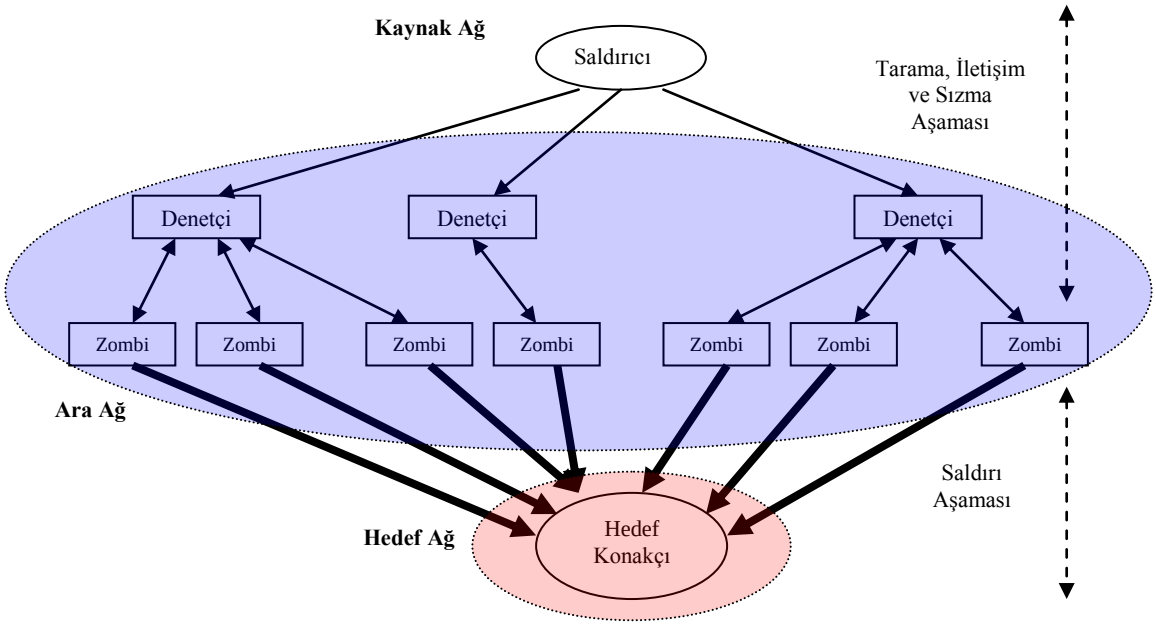
SYN taşkını saldırılarına karşı koymanın üç temel zorluğu vardır. Birincisi, Ping of Death veya Smurf gibi, DDoS saldırılarına sebep olan veri paketleri bir ateş duvarı tarafından durdurulabilmesine rağmen, SYN paketlerinin, normal trafiğin gerekli bir parçası olduğu için, süzülerek durdurulmalarının mümkün olmamasıdır. İkincisi, SYN paketleri küçük olduğu için, düşük bant genişlikli internet bağlantılarında bile çok sayıda gönderilebilmesidir. Sonuncusu, saldırıcıın, saldırdığı konakçıdan geri gelecek herhangi bir veriye ihtiyacı olmadığı için, saldırının gerçek kaynağını gizlemek ve süzülme imkânını ortadan kaldırmak amacıyla saldırı paketlerinin içine rastgele kaynak IP adresleri yerleştirmesidir [35].

Bir DDoS saldırısı Şekil 2.17'de görüldüğü gibi dört elemandan oluşmaktadır:

1. Gerçek saldırıcı;

2. Çok sayıda etmeni veya saldırıcıın "zombie" etmenlerle iletişimini sağlayan kanalları kontrol edebilen Denetçiler veya "master" konakçılar;
3. Hedef konakçıya paket akımı göndermekle sorumlu olan "daemon" etmenler veya "zombi" konakçılar;
4. Hedef konakçı.

DDoS savunma mekanizmaları, hedef, ara ve kaynak ağları olmak üzere üç farklı bölgeye yerleştirilebilmektedir. Hedef konakçıya olan trafiğin büyüklüğünün izlenmesine dayanan savunma yaklaşımlarının en büyük handikabı, DDoS saldırılarını ani ve yoğun olan normal trafikten ayırt edememeleridir. Bu trafik artışının anormal olup olmadığını anlayabilmek için biraz geç tepki verilmesi durumunda ise gerçek bir saldırıya uğrama riski doğmaktadır [36].



Şekil 2.17 : DDoS Saldırısı.

Aşağıda, DDoS saldırılarına verilebilecek teknik cevaplardan bazıları anlatılmaktadır:

- Filtreleme: Gelen trafik incelendiğinde, DDoS paketleri arasında benzerlik olduğu anlaşılacaktır. Çoğunlukla tek bir porta geliyor olabilirler veya paket büyüklüğü gibi belirli özellikleri aynı olabilir. Eğer bu şartlar sağlanıyorsa, yönlendiriciler, bu kriterleri sağlayan paketleri düşürmeleri için ayarlanabilir. Ne yazık ki, bu sadece bir geçici çözümdür, çünkü kötü niyetli kimseler saldırılarını farklı bir port, paket büyüklüğü veya filtre edilen herhangi bir

özellik için ayarlayabilirler. Ateş duvarlarının kullanılması ile UDP taşkını gibi istenilmeyen trafik hedef makinelere ulaşmadan engellenebilir. Yönlendiriciler "broadcast" adreslere hiç bir şey iletmemeleri yönünde ayarlanarak Smurf saldırılarının işi zorlaştırılabilir. Makinelerin "ping"lere ve "broadcast" mesajlara cevap vermesini engellemek de yardımcı olacaktır.

- Bant genişliğinin artırılması: Eğer bir web sitesi, ortalama miktarda bir bant genişliği veren bir "hosting" sağlayıcısına bağlı ise, bu bant genişliği, kolaylıkla orta büyüklükteki bir DDoS saldırısı tarafından tüketilebilir. Saldırının etkilerini soğurmak için, mevcut "hosting" sağlayıcısından veya daha geniş bir sağlayıcıdan daha fazla bant genişliği satın alınmalıdır. Eğer saldırıcılar artırılan bant genişliğini alt etmek için saldırıya daha fazla "bot" eklerse, bu çözüm de geçici bir çözüm olacaktır. Bazı firmalar, web sitelerin göçmesini engellemek için akıllı teknikler uygulamaktadırlar, fakat bunun karşılığında daha fazla para istemektedirler.
- IP adres değişikliği: Saldırı yapılan bilgisayarın IP adresi değiştirilerek saldırıdan kurtulunabilir. Ama ne yazık ki, saldırıların çoğu IP adreslerinden ziyade, alanları (domain) hedeflemektedir. IP adresi hedeflense bile, yeni IP adrsini hedeflemek çok kolay olacaktır.
- İsnat etme: Bir DDoS saldırısını durdurmanın en zor ve etkili yöntemi, arkasında kimin olduğunu bulmak ve yakalanmalarını sağlamaktır. Fakat, kötü niyetli kişilerin bu saldırılardan sorumlu tutuldukları ve ceza aldıkları durumlar çok yaygın değildir.

DDoS saldırılarında, hedef konakçının saldırıdan kurtulmak için alabileceği tedbirler kısıtlıdır. Saldırılar, güvenlik alanında yeteri kadar yatırım yapmadıkları için saldırılara açık olan savunmasız bilgisayarlar ordusu üzerinden yapılmaktadır. "Zombi" bilgisayar sahipleri, DDoS saldırısının riskleriyle yüzleşmedikleri ve saldırıya uğrayanların karşı-tedbir almalarını bekledikleri için, bu alanda yatırım yapmamaktadırlar. Kanada yasalarına gore bu tür zarar vermelerde tedbir almayıp da aracı olanlar da suçlu duruma düşmektedirler. Aynı yaklaşımın diğer ülkelere de yayılması sözkonusudur [4].

İnternet bağlantılı bilgisayar sahiplerinin DDoS saldırılarının başlatıcısı olarak kullanılmak amacıyla ele geçirilmesine engel olmak için alabilecekleri çok sayıda

tedbir vardır. Zayıflıkların düzenli olarak yamalanması, zararlı yazılımların bulaşıp bulaşmadığını kontrol etmek için düzenli taramalar, bilgisayarın DDoS ordusunda bir "zombi" olmasının önüne geçer.

2.1.4 Doğadan esinlenen algoritmalar

Zor optimizasyon problemlerini çözmenin alternatiflerinden birisi de, doğada gözlemlenen bazı temel prensiplere dayanan rastlantısal sezgisel yöntemleri kullanmaktır. Sızma Belirleme Sistemlerinin öğrenme aşamasında en çok kullanılan sezgiseller Evrimsel Algoritmalar ve Yapay Bağışıklık Sistemi'dir.

2.1.4.1 Evrimsel algoritmalar

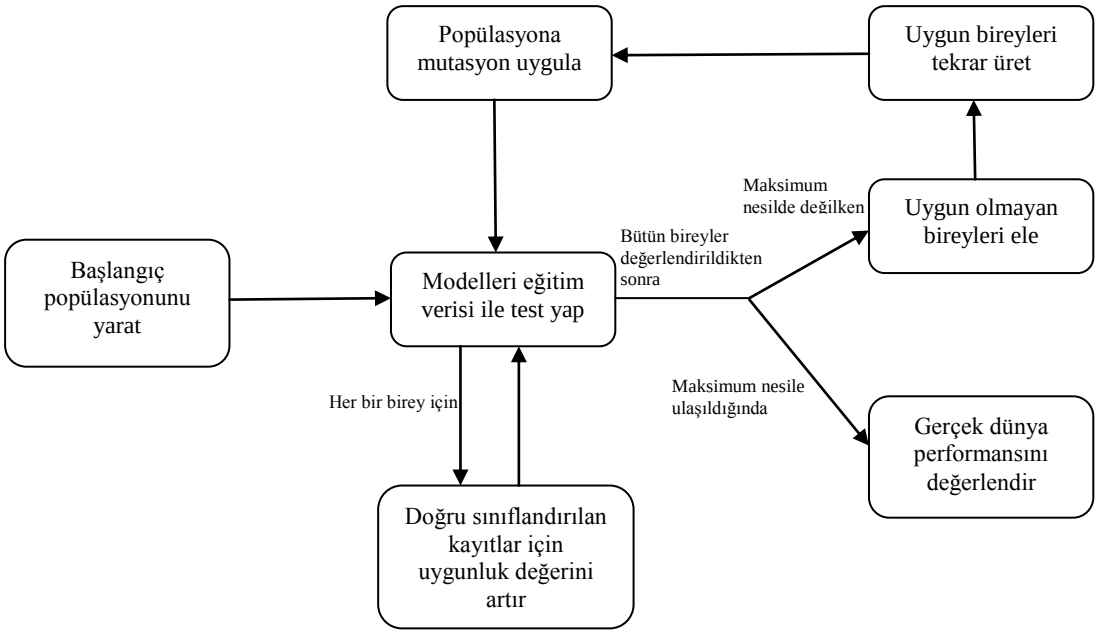
Evrimsel Algoritma, arama, optimizasyon, öğrenme gibi problemlere biyolojik popülasyon genetiği kurallarına dayanarak yaklaşan hesaplama tekniklerini içerir. Evrimsel algoritmalar formulize edilişlerine göre Genetik Algoritmalar, Evrimsel Programlama, Evrim Stratejileri ve Genetik Programlama gibi değişik isimlerle anılırlar.

Geleneksel arama metotlarında, bir çözüm adayı önerilir ve onu değiştirerek daha iyi çözümler elde etmeye çalışılır. EA'da ise, bir çözüm adayları popülasyonu oluşturulur ve bu populasyon zamanla evrimleşir. Bir adayın çözüme ne kadar yakın olduğu, uygulamaya bağlı bir fonksiyondur. Evrensellikleri, kullanım kolaylığı, paralel hesaplamalar için uygunluğundan dolayı EA'ların optimum çözümü bulması gradyant yöntemlere göre daha az zaman almaktadır [37].

Genetik algoritma, Darwin evrimine benzer olarak çalışan bir veri analiz yöntemidir. Bir bilgisayar simülasyonu içerisinde, her biri olası bir matematiksel modeli temsil eden çok sayıda bireyler topluluğu yaratılır. Her bir bireyin bir veya daha fazla kromozomu vardır. Bir birey, kromozomlarının toplam performansıyla ölçülür. Kromozomların tamamen rastgele düzenlenmesiyle başlangıç popülasyonu yaratılır; sonraki nesillerin bireyleri rastgele düzenlenirken aynı zamanda mutasyona uğrarlar. Darwinizm'de olduğu gibi, çok sayıda nesil sonrasında, performansı zayıf olan bireyler elenir ve performansı daha iyi olanların kopyalanmasına ve her bir nesilde kendilerini mutasyona uğratmalarına izin verilir.

Bir bireyin performansı uygunluk fonksiyonu ile ölçülür. Uygunluk fonksiyonu, Şekil 2.18'de gösterildiği gibi bir bireyin performansını, bireyin kromozomlarının

sonuçlarını problemin istenen sonuçları ile karşılaştırarak değerlendirir. Uygunluk, önceden belirlenmiş aralıkta en iyiden en kötüye olacak şekilde değerler alan bir ondalıklı sayı ile ifade edilir. Darwin evriminde olduğu gibi, düşük performans gösteren bireyler popülasyondan elenirler; yüksek performans gösteren bireyler klonlanırlar, mutasyona uğrarlar ve elenenlerin yerlerine geçerler. Mükemmel bir uygunluk skoru döndüren bir birey (ideal birey) bulunana kadar, biyolojik mutasyona benzer şekilde rastgele mutasyona uğrayan bireyler gelişirler ve yüksek kaliteli sonuçlar döndürürler. Eğer öyle bir birey bulunamazsa, genetik algoritma önceden tanımlanmış maksimum nesil sayısına ulaşılınca sonlanır.



Şekil 2.18 : Genetik Algoritma teorisi.

2.1.4.2 Çok-amaçlı evrimsel algoritmalar

Gerçek dünya problemleri genellikle birbiriyle çelişen birden fazla amacın eş zamanlı optimizasyonundan oluşur. Amaçların her birini sağlayan tek bir çözümün bulunması her zaman mümkün olmayabilir. Bu durumda problem bilgisine sahip olan karar vericiden, amaçların her biri için kabul edilebilir düzeyde olan alternatif çözümlerden seçim yapması istenir.

Bu çözümlerin her birine Pareto-Optimal çözüm; bu çözümlerin kümesine de Pareto-Optimal Çözümler Kümesi denir. Pareto-Optimal çözüm; amaçların herhangi biri için en kötü olmayan ve en azından bir amaç için diğerlerinden daha iyi olan

çözümdür. Diğer bir ifadeyle çözüm kümesindeki diğer herhangi bir çözüm tarafından bastırılmamış olan çözümdür [37].

Çok Amaçlı Evrimsel Algoritma, aşağıdaki iki durumun değerlendirildiği bir EA uzantısıdır [39]:

- Üzerinde-baskınlık-kurulmayan çözümlerin üzerinde-baskınlık-kurulanlara tercih edileceği şekilde bireylerin seçilmesi,
- Popülasyonda Pareto optimal kümesinin elemanlarından olabildiğince fazla bulundurabilecek şekilde çeşitliliğin korunması.

2.1.4.3 Yapay bağışıklık sistemi

Bağışıklık sistemi, yabancı cisimlere karşı organizmanın savunmasından sorumlu olan hücreler ve organlardan oluşan karmaşık bir ağdır. Bağışıklık sisteminin temel özelliklerinden birisi kendine-ait (self) ve kendine-ait-olmayan (non-self) molekülleri ayırt edebilme kabiliyetidir. Organizmanın her hücresi, yapısında, kendine-ait moleküller olarak tanımlanan moleküllere sahiptir. Diğer yandan, yabancı organizmaları oluşturan moleküller kendine-ait-olmayan moleküller olarak tanımlanırlar. Bağışıklık sistemini oluşturan organlar lenfoid olarak adlandırılırlar, çünkü lenfositlerin gelişimi ve idamesinden sorumludurlar. B-hücreleri ve T-hücreleri lenfositlerin iki ana sınıfıdır. Organizma bir hastalığa yakalandığında, bu hücrelere aktive edilirler. Hastalık yenildiğinde, bu hücrelerden bazıları hastalığın hafızasını tutarlar. Böylece, organizma aynı antijenle başka bir ortamda karşılaşırsa, bağışıklık sistemi onu tanıyacak ve yok edecektir. İnsanların bağışıklık sisteminde pasif doğal bağışıklık vardır; bebekler hayatlarının ilk aylarında annelerinden aldıkları antikorlar ile korunurlar. Plasenta da hareket eden IgG antikorları, annenin bağışık olduğu mikroplara karşı bebeği de bağışık kılar. Buna ek olarak, çocuklar annelerinin sütlerinden IgA antikorları alırlar [40].

Bağışıklık hücreleri, kemik iliği ve timusta oluşurlar, olgunlaşmamış bağışıklık hücrelerinden olgunlaşmışlara ve sonra hafıza bağışıklık hücrelerine doğru evrim geçirirler. Bağışıklık evrimi sürecinde, uyum yeteneği fazla olan yeni hücreler, uyum yeteneği kötü olan hücrelerin yerine geçebilmektedir. Bağışıklık hücrelerinin yaşam döngüsünde kendinden olanlara-tolerans, klon seleksiyon, mutasyon, hafıza ve ölüm işlemleri yer almaktadır.

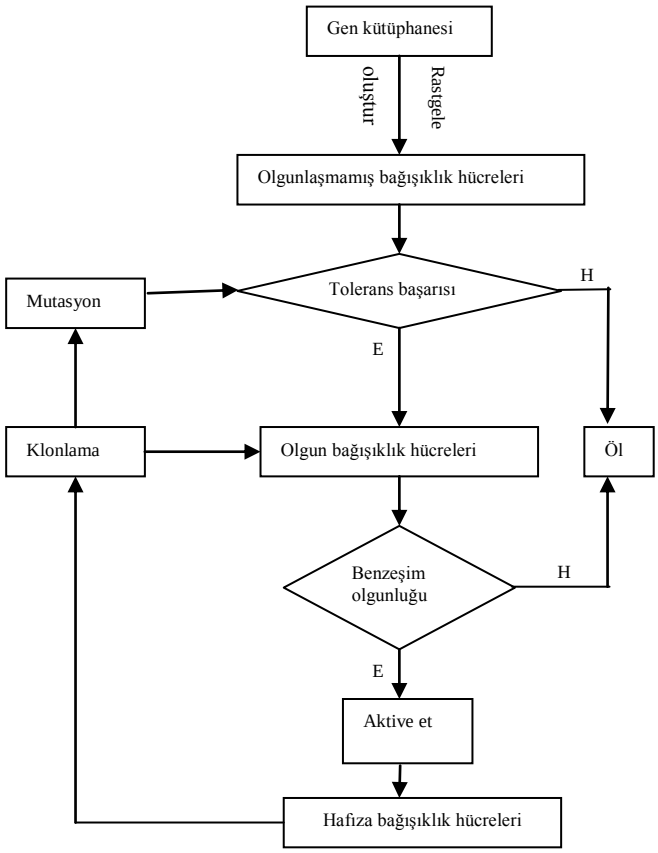
Bağışıklık hücreleri, Şekil 2.19'da görünen hayat döngüsündeki görevleri ve bulundakları aşamalara göre, olgunlaşmamış hücreler, olgun hücreler ve hafıza hücreleri olarak üç tür hücre içerirler. Bağışıklık hücreleri başlangıçta rastgele yaratılan olgunlaşmamış hücreler halindedir. Eğer olgunlaşmamış bağışıklık hücresi Negatif Seleksiyon sürecinde kendine-ait hücreleri tanımazsa, olgun bağışıklık hücresi olacak şekilde evrim geçirir. Olgun bağışıklık hücresi, tanıdığı antijen miktarı belli bir eşik değerini geçerse hafıza bağışıklık hücresi olacak şekilde evrim geçirir. Olgun ve hafıza bağışıklık hücrelerinin ikisi de antijenleri tanırlar. Aralarında iki fark vardır: birincisi, olgun hücrenin ömrü sonludur ama hafıza hücresininki sonsuzdur; ikincisi, olgun hücrenin aktive olma eşiği hafıza hücresininkinden daha yüksektir. Hafıza bağışıklık hücresi bir adet kendine-ait-olmayan tanıdığında aktive olurken, olgun bağışıklık hücresi tanıdığı kendine-ait-olmayan miktarı belli bir eşiği geçince aktive olur. Hafıza bağışıklık hücresi bilinen antijenleri belirleyebilirken, olgun bağışıklık hücreleri yeni bilinmeyen antijenleri belirleyebilmektedir [41].

Negatif seleksiyonun amacı kendine-ait hücrelere tolerans tanımaktır. Bu durum, bağışıklık sisteminin kendine-ait hücrelere tepki vermezken bilinmeyen antijenleri belirleme kabiliyeti ile ilgilidir. T-hücreleri yaratılırken, alıcılar (receptor) rastgele oluşturulurlar. Sonra, timus içerisinde negatif seleksiyon denilen bir sansürleme sürecinden geçerler. Burada, kendine-ait proteinlere tepki veren T-hücreler yok edilir, sadece kendine-ait proteinlere bağlanmayanların timusu terk etmesine izin verilir. Olgunlaşan T-hücreleri bağışıklık fonksiyonlarını yerine getirmek ve vücudu yabancı antijenlere karşı korumak için vücutta dolaşırlar.

Klonlama seleksiyon prensibi, antijen profilini tanımayanların değil sadece tanıyanların seçileceği düşüncesini ortaya koyar. Bir antikorla bir antijen güçlü bir şekilde eşleşiyorsa, ilgili B-hücresi kendi klonlarını üretmek ve sonra daha fazla antikor üretmek üzere harekete geçirilir [42].

Her bir bağışıklık hücresi, bütün vücutta dolaşan ve yabancı mütecavizlere karşı vücudu savunan özerk gezgin etmenler olarak düşünülebilir. Sızma belirlemenin öncelikli amacı konakçının ve ağın normal davranışlarını bozuk davranışlardan ayırt etmektir. Bu, kendine-ait ile kendine-ait-olmayanı ayırt eden biyolojik bağışıklık sisteminin kendini-koruma mekanizmasına benzemektedir. Ağ veya konakçının normal davranışları kendine-ait ve bozuk daranışları kendine-ait-olmayan olarak

değerlendirilebilir. Belirleyiciler (dedector) kendine-ait olmayanı temsil eder ve negatif seleksiyon sırasında olgunlaşırlar [44].



Şekil 2.19 : Bağışıklık hücrelerinin hayat döngüsü.

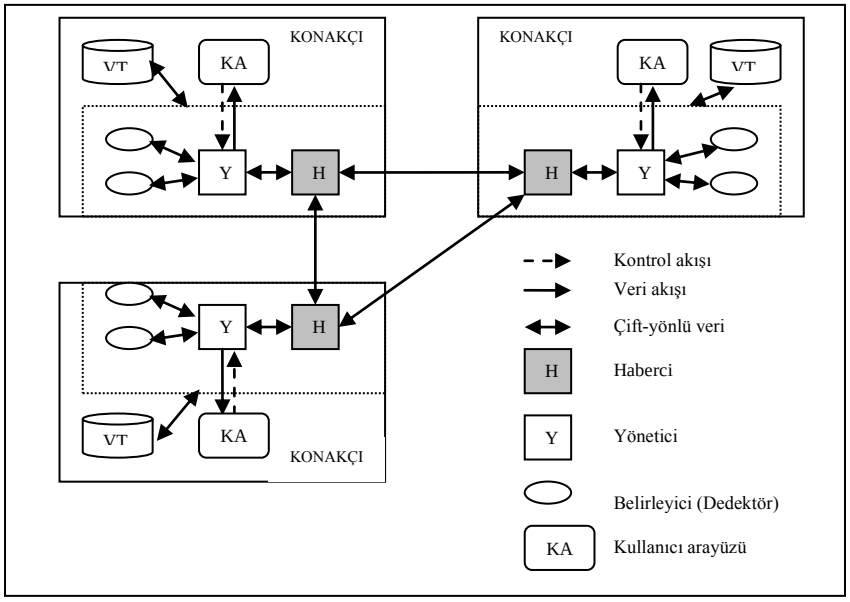
2.2 İlgili Çalışmalar

Bu konuda yapılan çalışmalar, Dağıtık SBS ve Anomali-temelli SBS başlıkları altında ayrı ayrı ele alınacaktır.

2.2.1 Dağıtık SBS

IADIDS [6] çalışmasında, tekil nokta olan merkezi ünitenin çökmesi problemini çözmek için, bağımsız etmenleri temel alan dağıtık bir SBS ortamı önerilmektedir. Sistemdeki uygulama, birbirine bağlı etmen varlıklarından oluşmaktadır ve istenildiğinde ana parçalara dokunmadan yeni etmenler eklenebilmektedir.

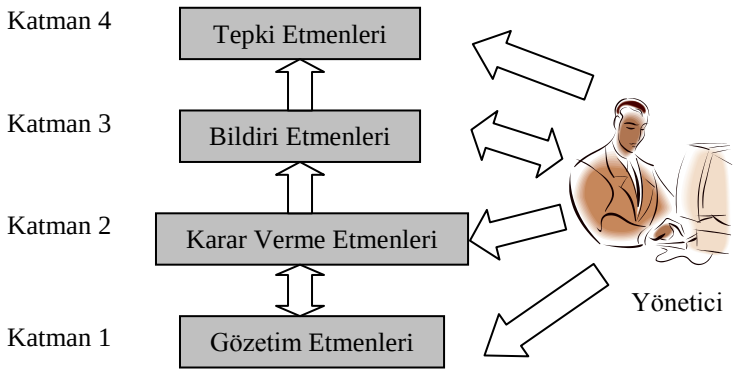
Aynı konakçının bütün varlıkları hiyerarşik yapı organizasyonu içerisindedir. Bu varlıklar arasında, yönetici en yüksek seviyede, belirleyici daha aşağı seviyededir. Şekil 2.20'de görüldüğü gibi, değişik makineler arasında yardımlaşan varlıklar eşit pozisyondadır ve bu varlıklar arasında kontrol merkezi yoktur. Bu yapıdaki etmenler, eşdüzeyler, üst ve alt düzeyler ile DARPA ve "Initiative External Interfaces Working Group" adlı çalışma grubu tarafından geliştirilen "Knowledge Query and Manipuliation Language (KQML)" denilen etmen iletişim dili ve protokolü ile haberleşirler ve işbirliği yaparlar. Sistem, bir prototip olarak ele alınmış olup uygulama safhasına geçilemediği belirtilmiştir.



Şekil 2.20 : Dağıtık yapı [6].

[45] numaralı çalışmada anlatılan SBS, geniş ve tek parça bir modül yerine, otonom gezgin etmenleri temel alan modüler bir yaklaşım önermektedir. Sistem, etmen katmanlarından oluşmaktadır. Her bir katman bir üst katmanına bilgi göndermektedir. Gezgin etmenler saldırıya-özel değildir ve veri analizi yapmazlar. Veri analizi, karar verici etmenler tarafından yapılmaktadır. Bir etmen, şüpheli bir hareket görürse derhal o hareket hakkında sistemin diğer etmenlerini haberdar eder. Hemen sonra, ilgili sızma adayı türünde daha yüksek uzmanlık derecesine sahip bir ya da daha fazla etmen aktive edilir.

Şekil 2.21'de görüldüğü gibi, Katman N, Katman N-1'in sunduklarını kullanır, görevini yapar ve Katman N+1'e hizmet sunar. Gözetim Etmenlerinin topladığı bilgilere dayanarak, Karar verme Etmenleri harekete geçer ve olası sızmaları belirler ve analiz eder. Eğer bu etmenler bir hareketi şüpheli olarak değerlendirirlerse, ağ yöneticisini haberdar eden (e-posta, telefon, alarm, vb.) veya daha üst seviyedeki etmenleri aktive eden Bildiri Etmenlerini harekete geçirir. En üstte, bildiri etmenlerinden aldıkları bilgilere dayanarak olası sızmalara otomatik olarak karşı saldırıda bulunan veya ağ yöneticisinin komutuyla aktive olan Tepki Etmenleri yer alır. Etmenlerin tek başlarına sızmaları belirleme otoritesi yoktur. Bu karar ancak birden fazla etmenin anlaşmasıyla alınabilmektedir. Eğer bir sızmadan sadece bir etmen şüpheleniyorsa, bu şüphe için yapılacak oylama sonucunda diğer etmenler tarafından bu şüphe bertaraf edilebilir. Aglet [20] etmen platformunun kullanıldığı belirtilen bu çalışmada uygulama ile ilgili sonuçlar verilmemiştir.



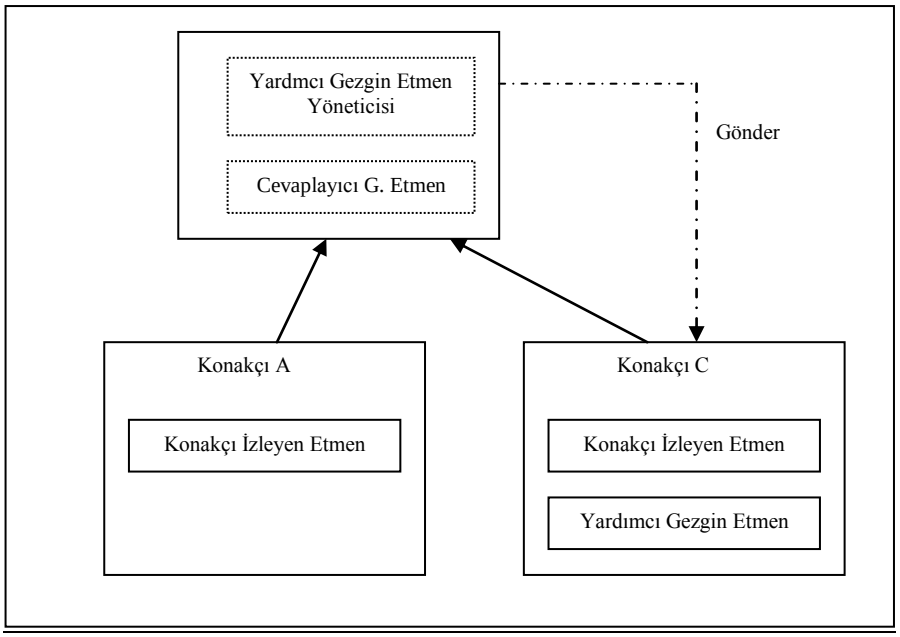
Şekil 2.21 : Katman Modeli [45].

[46] numaralı çalışmanın yönetici bileşeni, sisteme saldırıya-özel olmayan ve analiz edilmek üzere sızma verisi toplayan gezgin etmenler yollamaktadır. Sabit etmenler konakçıları izlemek için kullanılırken, devriye gezen etmenler yalnızca konakçılardan sızma ile ilgili veri toplamaktadır. Bu verileri ilişkilendirme ve bir karar verme işlemini yapmamaktadırlar. Çalışmada, GPSY [26] gezgin etmen platformu kullanılmış olup uygulama sonuçları verilmemiştir.

Sistemin Şekil 2.22'de görüldüğü gibi 4 adet bileşeni vardır: Yönetici (Manager), Yardımcı Gezgin Etmen (Assistant MA), Cevaplayıcı Gezgin Etmen (Response MA), Konakçı İzleyen Etmen (Host Monitoring Agent). Ağdaki izlenen her bir konakçıya, yerel sızma belirleme yapması için bir adet Konakçı İzleyen Etmen yüklenmiştir. Sistemin güvenliği ve esnekliği için bütün bileşenler gezgin etmenler

içine konmuşlardır. Her bir konakçıya Gezgin Etmen Sisteminin çalışması ve taşınması için çalıştırma ortamı oluştursun diye Gezgin Etmen Platformu kurulmuştur. Sistemin çalışma süreci aşağıda açıklanmıştır:

1. Bir konakçıda sızma belirlenebilirse, Konakçı İzleyen Etmen doğrudan Yöneticiye uyarı gönderir. Aksi takdirde, Konakçı İzleyen Etmen Yöneticiden yardım ister ve şüpheli hareketi kayıt altına alır.
2. Yönetici yardım isteğini alınca, değişik konakçılardaki şüpheli hareketler birleştirilip bir dağıtık sızma belirlenebilir mi diye karar vermek için bilgi toplamak amacıyla, izlenen diğer konakçılara bir adet Yardımcı Gezgin Etmen gönderir.
3. Yardımcı Gezgin Etmen döndükten sonra, Yönetici toplanan bilgileri analiz eder ve dağıtık sızma tanımlaması yapar. Eğer dağıtık sızma tesbit edilirse, Yönetici, gerekli karşılığı vermek için bir adet Cevap Gezgin Etmen gönderir.



Şekil 2.22 : Sistem Mimarisi [46].

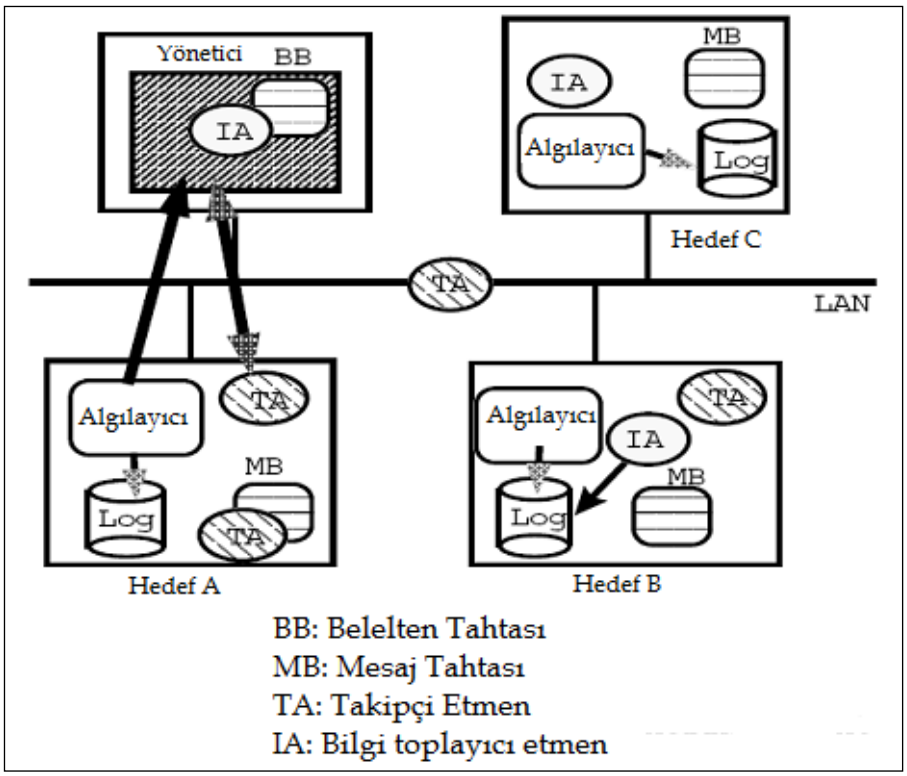
Konakçı İzleyen Etmen içerisinde ağ bağlantısını, dosya işlemlerini ve öncelik işlemlerini izlemekten sorumlu üç adet alt-etmen vardır. Sızma belirleme sürecini birbirleriyle koordineli olarak tamamlarlar. Detaylar aşağıda açıklanmıştır:

İlk başta, ağ alt-etmeni paket başlığı, geliş zamanı ve büyüklüğünü de içerecek şekilde ağ veri paketlerini analiz eder. Sonrasında, eğer bir problem ortaya konabilirse, Yorumlama Tabanındaki ilgili Yorumlama Ağacına başvurulur. Yorumlama Ağacı, bu problemi analiz eder ve bir sonuca varır. Bu yorumun şüphe seviyesi alınarak, ilgili erişimin süphe seviyesi artırılır. Eğer bu erişimin şüphe seviyesi belirli bir eşik değerin (bu çalışmada eşik değer 0.8'dir) üstüne çıkarsa, sızma olduğuna kanaat getirilir. Konakçı İzleyen Etmen, Yönetici'ye bir alarm mesajı gönderir ve sızmaya karşı-cevap (bağlantının kesilmesi gibi) verir. Aksi takdirde, ağ alt-etmeni bu şüphe seviyesini diğer alt-etmenlere yönlendirir. Eğer diğer alt-etmenler bu erişimi diğer sistem kaynakları açısından şüpheli bulurlarsa, erişimin şüphe seviyesi tekrar artırılır.

Konakçı İzleme Etmeni, konakçı ve kullanıcı bilgilerinin kayıtlarını tutar, Yorumlama Tabanındaki ilgili yorumlama ağacını günceller, konakçı ve kullanıcının şüphe ağırlığını değiştirir. Bu yöntemle, birçok etmen yerel sızma belirleme ve otomatik öğrenme işlemini beraber yaparlar.

IDA (Intrusion Detection Agent) [47] çalışmasında, gezgin etmenler sızıcıları izlerler, sızma rotası boyunca yalnızca sızma ile ilgili bilgileri toplarlar ve gerçekten bir sızma oluşup oluşmadığı kararını verirler. Bu işlemler sayesinde etkin bilgi toplama yapılabilmekte ve kötüye kullanılan ara konakçılar da tesbit edilebilmektedir.

IDA, kullanıcıların bütün aktivitelerini analiz etmek yerine, "Marks Left by Suspected Intruder (MLSI)" denilen sızmalarla ilgili olabilecek olayları izler. Eğer bir MLSI bulursa, bu MLSI ile ilgili bilgileri toplar, analiz eder ve bir karar verir. IDA Şekil 2.23'de görüldüğü gibi, bir yönetici, algılayıcılar, belleten tahtaları, mesaj tahtaları, takipçi etmenler ve bilgi toplayan etmenlerden oluşmaktadır ve D'Agent [23] etmen platformunu kullanmaktadır.



Şekil 2.23 : IDA yapısı [47].

Aşağıda, bir algılayıcı hedef sistemde bir MLSI belirlediğinde IDA'nın nasıl çalıştığı açıklanmaktadır:

1. Hedef sistemdeki her bir algılayıcı sistem kayıtlarında bir MLSI arar.
2. Eğer bir algılayıcı bir MLSI belirlerse, bu durumu yöneticiye rapor eder.
3. Yönetici, MLSI belirlenen hedef sisteme bir takipçi etmen gönderir.
4. Takipçi hedef sisteme ulaşır ve bir bilgi toplayıcı etmeni aktive eder.
5. Bilgi toplayıcı etmen, hedef sistemde MLSI ile ilgili bilgi toplar.
6. Takipçi etmen, bilgi toplayıcı etmeni aktive ettikten sonra, MLSI'ın orjinal noktasını araştırır. Uzak kullanıcının kimliğini, sistemde koşturan prosesler ve ağ bağlantısı hakkında toplanmış oaln bilgilerden çıkarmaya çalışır.
7. Bilgi toplayan etmen, bilgileri topladıktan sonra takipçi etmenden bağımsız olarak yöneticiye döner ve belleten tahtasına bilgileri işler.
8. Takipçi etmen, takip rotasındaki diğer hedef sisteme taşınır ve yeni bir bilgi toplayıcı etmeni aktivite eder.

9. Takipçi etmen, eğer rotanın başına ulaşırsa, başka bir yere taşınamıyorsa veya izleyeceği rotayı diğer takipçi etmenler izlediyse yöneticiye geri döner.

[18] numaralı çalışmada, merkezi koordinatörü olmayan bir eşdüzeyler-arası SBS önerilmektedir. Komşuların birbirlerini kolaçan ettikleri bir sanal komşuluk ortamı yaratılmıştır. Her bir site, düzenli olarak komşularını ziyaret ve kontrol etmek için gezgin etmen gönderir. Çelişkili veya olağan dışı bir davranış gözlemlenirse, gözlemci komşu, tehlike altındaki siteye karşı harekete geçmek için bir oylama başlatır. Her bir site, kritik bilgilerini komşularında saklar. Bu bilgiler, kritik veri dosyalarının, işletim sistemi dosyalarının, sistem ikilik dosyalarının sağlamaları ve erişim kontrol dosyaları, konfigürasyon dosyaları, normal MIB aktiviteleri dosyalarıdır. Aynı zamanda, komşusunun komşularının listesi de vardır.

Buradaki konsept, gerçek hayatta yaşananları bazı yönleriyle taklit eden bir SBS yaratmaktır. Gerçek hayatta da komşuluk kültürü vardır. Kapılar kilitlenir ve kıymetli eşyalar korunur. Komşular kolaçan edilirken aslında kişiler kendilerini de korurlar. Çok rahatsız edici olunmadan, koşunun evinin etrafında bir yabancı veya şüpheli bir olayın gerçekleştiği görüldüğünde, komşular haberdar edilir. Komşunun kapısı açık bulunursa veya başkasının komşunun evine girmeye çalıştığı görülürse hemen bir alarm verdirilir.

Bu SBS'de bir Şef, Detektifler ve Polisler vardır. Polisler gezgin etmenlerdir. Farklı yerlere gönderilebilecek şekilde çok sayıda, farklı tipte göreve-özel polis vardır. Gönderildikleri yerlerde çalıştırılırlar ve dedektiflere rapor verirler. Detektifler, polisleri koordine etmek ve farklı komşulara yollamak ile sorumludurlar. Dedektif, polislerden aldığı tüm raporları analiz eder ve eğer şüpheli bir durum görürse, Şefi haberdar eder. Gezgin etmenler Perl kullanılarak kodlanmıştır.

IDReAM [17], gezgin etmenleri doğadan-esinlenen öğrenme algoritmalarıyla birleştiren bir çalışmadır. SBS yapay bağışıklık sistemini kullanırken, Sızma Cevaplama Sistemi (SCS) karınca kolonilerinin mekanizmasını kullanır. Gönderilen her gezgin etmen, değişik düğümler arasında rastgele dolaşır. Toplanan bilgiler arasındaki ilişki, önceki gezgin etmenler tarafından bırakılan depozitlerin yardımıyla kurulur. IDReAM'in yapısında, her düğümün üzerinde koşan J-Seal2 [27] adı verilen bir gezgin etmen platformu vardır.

Bir Sızma Belirleme Etmeninin (IDA) yaptığı işlemler aşağıda sıralanmıştır:

- Rastgele olarak hareket eder. Bir makineye geldiğinde, diğer IDA'lar tarafından bırakılan bilgileri ve önceden ezberlediği bilgileri dikkate alarak gelen olayları yoklar ve Şüphe Endeksi'ni (SI) hesaplar.
- Eğer SI eşik değerini geçerse, bir alarm oluşturur, bir feromon geliştirir ve yayar.
- Yeni bir normalden sapma belirlemek için dolaşmasına devam eder.
- Kendisini yaratan AgentHost'a dönünce, yasaklı olayların kendinden-olmayan profilindeki SI'yı günceller.

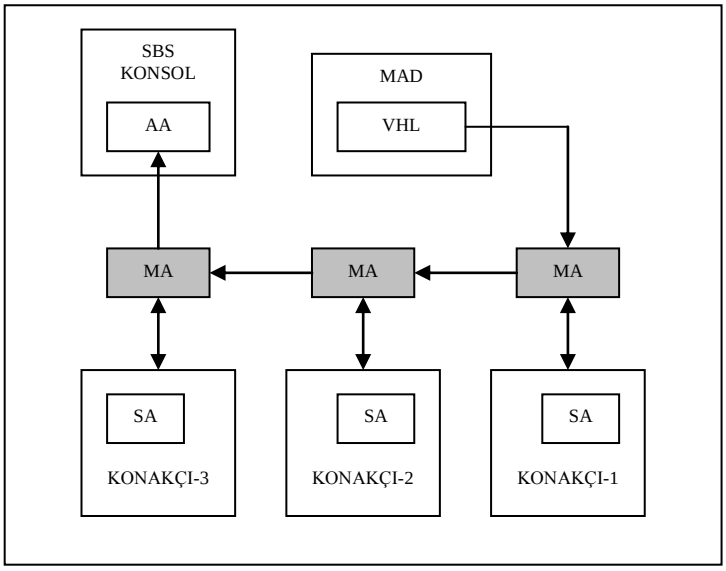
Bir Sızma Cevaplama Etmeninin (IRA) yaptığı işlemler aşağıda sıralanmıştır:

- Normal durgun durumunda, rastgele dolaşır ve ağda feromon bilgisi arar.
- Eğer bir feromon bulursa, takip durumuna geçer ve feromonu kaynağına kadar takip eder. İzi takip ettikçe, iz üzerindeki düğümlerdeki feromonların buharlaşmasını hızlandırır.
- Alarmın kaynağına ulaştığında, orjinal saldırıya bir cevap verir ve durgun durumuna geçer. Başka bir feromon bulmak için rastgele yürüyüşüne devam eder.

DIDMA [3] çalışmasında, durağan etmenlerden aldıkları sızma ile ilgili verilerin birleştirilmesi ve ilişkilendirilmesi görevini yapan gezgin etmenler kullanılmaktadır. DIDMA, bu tez çalışmasında önerilen ikinci yöntemin temelini oluşturmaktadır. Gezgin etmenler, veri analiz hesaplamalarını sızma verisinin bulunduğu yere getirerek ağ bant genişliği kullanımını azaltmaktadır. Bu çalışmada, Voyager [24] gezgin etmen platformu kullanılmıştır.

DIDMA'da, durağan ve gezgin etmen diye adlandırılan iki adet yazılım elemanı kullanılmıştır. Durağan etmenler, konakçılarda bulunurlar ve üzerinde şüpheli aktivite belirlenen konakçıları göstermekle sorumludurlar. Gezgin etmenler, durağan etmenlerin, üzerinde şüpheli aktivite belirledikleri bütün konakçıları incelerler. Gezgin etmen gittiği her konakçıdaki durağan etmenden verilerin saldırı izi taşıyan kısmını alır ve aynı tipte şüpheli aktivite belirleyen diğer durağan etmenlerden aldığı verilerle ilişkilendirir. Sonuçta oluşan veri, ziyaret edilen bir sonraki konakçıya taşınır ve aynı işlemler tekrarlanır. Bu teknik sayesinde, dağıtılmış veri analizi yapılarak SBS daha ölçeklenebilir hale getirilir. Gezgin etmenlerin taşıdığı küçük

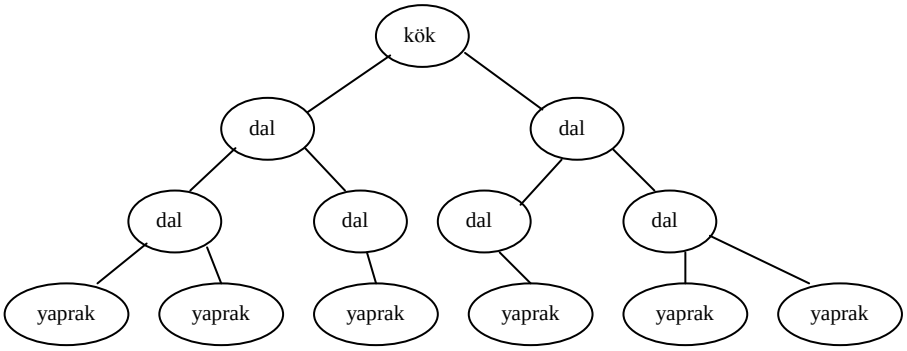
kodlar ve saldırı takip verileri, ağ üzerinden gönderilen büyük, işlenmemiş verilerden daha az yüke neden olur. Sistemin, Şekil 2.24'de görüldüğü gibi, Durağan Etmenler (SA), Gezgin Etmenler (MA), Gezgin Etmen Göndericisi (MAD), Ziyaret Edilen Konakçı Listesi (VHL), Alarm Veren Etmen (AA) ve SBS Konsolü bileşenleri vardır.



Şekil 2.24 : Sistemin dağıtık yapısı [3].

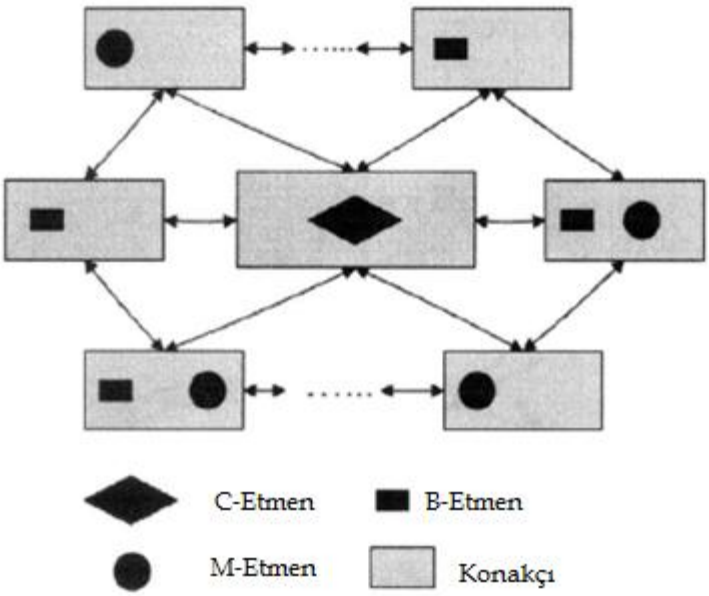
LDIDS [43] çalışmasında yer alan hiyerarşi, yaprak düğümleri, dal düğümleri ve bir kök düğüm içerir. Yaprak düğümler, yerel ağ aktivitelerini izlerler; dal düğümleri, her bir çocuk düğüm ağını izlerler; kök düğüm, bütün ağın aktivitelerini izler.

Şekil 2.25'deki yaprak düğümler, log kayıtları, ağ veri blokları ve diğer güvenli bileşenler tarafından gönderilen alarmları içeren yerel bölge verilerini toplar. Veriler, uygun analiz yöntemleri ile analiz edilirler. Sonrasında, yerel SBS, sızma olaylarına göre uyarılar üretir ve bu uyarıların bazılarına yerel olarak karşılık verir. Yerel SBS, yerel olarak analiz edemediği uyarı ve verileri anne düğümüne gönderir. Anne düğüme gönderilen bu uyarı ve verilerin diğer bölümlerdekilerle ilişkisi olabilir. SBS, çocuk SBS'lerden uyarı ve verileri alır, çocuk düğümlerde sızma olup olmadığına karar verir. Bütün SBS'ler birbirleriyle yardımlaşırlar. Eğer çocuk düğümde potansiyel bir tehlike bulurlarsa, diğer düğümleri haberdar eder ve tehlikeyi bertaraf etmek için bazı önlemler alırlar.



Şekil 2.25 : Sistemin mimarisi [43].

IA-DIDM [44] çalışmasında biyolojik bağışıklık sistemindeki lenfositlerin çalışma mekanizmasından esinlenilerek, sızma etmenlerini temel alan yeni bir dağıtık sızma belirleme modeli önerilmektedir. Modelin iç ağ yapısı Şekil 2.26'da görülmektedir. Bağışıklık etmenleri, işlevlerine gore merkezi control etmeni (C-Agent), B-Agent ve hafıza etmeni (M-Agent) olarak üç farklı sınıfa ayrılmaktadır. C-Agent sunucuda yer alırken, B-Agent ve M-Agent konakçılar arasında dolaşır ve yabancı saldıranları gözlemlerler.



Şekil 2.26 : IA-DIDM yapısı [44].

Bağışıklık etmenleri, bir veya daha fazla belirleyici taşırlar. Başlangıçta, C-Etmen, negatif seleksiyon sonrası olgun belirleyiciler halini alacak olan olgun-olmayan belirleyiciler kümesini rastgele olarak üretir. Sonra, C-Etmen, belirleyicileri taşımaları için B-Etmenleri üretir. Her bir B-Etmen farklıdır ve değişik belirleyiciler taşırlar.

Çizelge 2.3 : Dağıtık SBS çalışmalarının karşılaştırılması.

S/N	Çalışmalar	Avantajlar	Dezavantajlar
1	[18]	Tüm gezgin etmenleri yaratan ve kontrol eden merkezi bir koordinatör bulunmamaktadır.	Her bir bölgenin analizi, önceden gelen raporları ve gezgin etmenleri koordine eden sabit etmenler tarafından yapılmaktadır.
2	[47]	Gezgin etmenler sadece sızmalarla ilgili bilgi toplamak için hedef sistemlere kendiliğinden giderler, böylece gereksiz sistem kayıtları analiz sunucusuna taşınmamış olur.	Sızma şüphesi hakkındaki karar, merkezi olarak verilmektedir. Takipçi gezgin etmenler sızmanın yolunu takip etmek ve çıkış noktasını belirlemek için diğer konakçıları dolaşırlar; fakat bu takip ancak bir konakçıya yapılan sızmanın ağdaki başka bir konakçıdan zincirleme olarak gelmesi durumunda işe yarayacaktır ki, çoğu zaman bu geçerli değildir.
3	[17]	Gezgin etmenler, konakçılar arasında rastgele dolaşmaktadırlar ve toplanan bilgiler arasındaki ilişki önceki gezgin etmenler tarafından bırakılan izler kullanılarak dağıtık olarak belirlenmektedir. Gezgin etmenlerin gezi yolları her bir konakçıda dinamik olarak düzenlenmektedir.	Gezgin etmenlerin ne zaman, kaç tane, kim tarafından yaratıldığı konularında yeterli bilgi bulunmamaktadır.
4	[45]	Gezgin etmenler mesajlaşarak şüpheli durumları ve alınacak tedbirleri birbirleriyle paylaşmaktadır. Her bir etmen sistemin sadece küçük bir kısmını gözlemlemektedir.	Gezgin etmenler saldırıya özel değil ve veri analizi yapmamaktadır. Veri analizi, ayrı karar verici etmenler tarafından yapılmaktadır.
5	[46]	Gezgin etmenlerin ve Yöneticinin zaman sayaçları eşzamanlı çalışmaktadır. Sayaç sıfırlanınca, görevinin bitmesine bakmaksızın gezgin etmenler Yöneticiye dönmek zorundadır. Aksi durumda, aynı görevli başka bir etmen daha yaratılır.	Gezgin etmenler sadece bilgi toplamaktadır, analiz yapmamaktadır.
6	[3]	Gezgin etmen yaratıcısı çalışamaz duruma gelse dahi yaratılmış olan gezgin etmen görevini tamamlayabilmektedir.	Gezgin etmenler merkezi olarak yaratılmakta ve listelerinde belirtilen yolu takip etmektedirler. Sabit etmen tarafından gönderilen saldırı şüphesi mesajı, ancak konakçıdan kaynaklanan bir DoS saldırısı gerçekleştiğinde yaratılmaktadır.
7	[6]	Tekil nokta olan merkezi ünitenin çökmesi problemini çözmek için, bağımsız etmenleri temel alan dağıtık bir SBS ortamı önerilmektedir. Değişik makinelerde bulunan ve birbirleriyle yardımlaşan varlıklar arasında kontrol merkezi yoktur.	Gezgin etmenler sadece bilgi toplamaktadır. Sistem, bir prototip olarak ele alınmış olup uygulama safhasına geçilemediği belirtilmiştir.
8	[43]	Bütün SBS'ler birbirleriyle yardımlaşırlar. Eğer çocuk düğümde potansiyel bir tehlike bulurlarsa, diğer düğümleri haberdar eder ve tehlikeyi bertaraf etmek için bazı önlemler alırlar.	Hiyerarşik karar ağacı yapısı, merkezi tek-nokta zaafiyetine neden olabilir.
9	[44]	Yapay bağışıklık sisteminde yer alan b-agent ve m-agent (hafıza) etmenleri ağda dolaşarak yabancı paketleri gözlemlerler. Hafıza etmenleri daha hızlı belirleme yapar.	Merkezi sunucuda yer alan C-Etmen bütün etmenlerin raporlarını değerlendirir ve son kararı verir.

Her bir B-Etmenin planlanan bir gezi yolu vardır. Gezi yollarının tasarımında etmenlerin düzenli dağılımına dikkat edilir. Her bir B-Etmen kendi gezi yoluna göre ağda dolaşmaya başlar, şüpheli davranışları izler ve kararını C-Etmene rapor eder. C-Etmen bütün etmenlerin raporlarını değerlendirir ve son kararı verir.

Her belirleyicinin aktivasyon eşiği (a), uyarı eşiği (w), benzerlik olgunlaşması eşiği (m) olarak üç eşik değeri vardır ($a < w < m$). Eğer şüpheli bir sızma hareketi ile belirleyici arasındaki benzerlik derecesi a'dan fazlaysa, belirleyici aktive edilir; eğer benzerlik w'den fazlaysa, bu şüpheli sızma hareketi gerçek bir sızmadır kararı verilir; eğer benzerlik m'den fazlaysa belirleyici o sızma hareketiyle ilgili olarak olgunlaşmış bir hafıza belirleyicisi halini alır.

Çizelge 2.3'de Dağıtık SBS çalışmalarının temel avantaj ve dezavantajlarının ele alındığı karşılaştırma tablosu yer almaktadır.

2.2.2 Anomali-temelli SBS

Kalle Burbeck, "ADWICE, Anomaly Detection With fast Incremental Clustering " [8] adlı "kümeleme" çalışmasında anomali belirleme sistemlerinde kullanılan öğrenme yöntemlerini şu başlıklar altında toplamıştır:

- Kümeleme
- Bayes Öğrenme
- Yapay Sinir Ağları
- Sonlu Otomata
- Yapay Bağışıklık Sistemi
- Karar Ağaçları
- Olasılıksal Model
- Gizli Markov Modeli
- İstatistiksel Model

ADWICE çalışmasında, sınıflandırma-temelli yöntemlerin eğitim verisi ihtiyacını azaltmak ve denetimsiz (unsupervised) anomali belirlemenin saldırı hacmi varsayımına gerek duymamak için, eğitim verisinin sadece normal veri içerdiği saf anomali belirleme kullanılmaktadır. Eğer "saldırıların veri içerisinde seyrek olduğu"

varsayımı tutarsa, saldırılar küme büyüklüğüne göre belirlenecektir; küçük kümeler saldırı verisi olacaktır. Bu çalışmada ayrıca, sızma belirleme performansını artıran bir grid indeks yapısı da önerilmektedir.

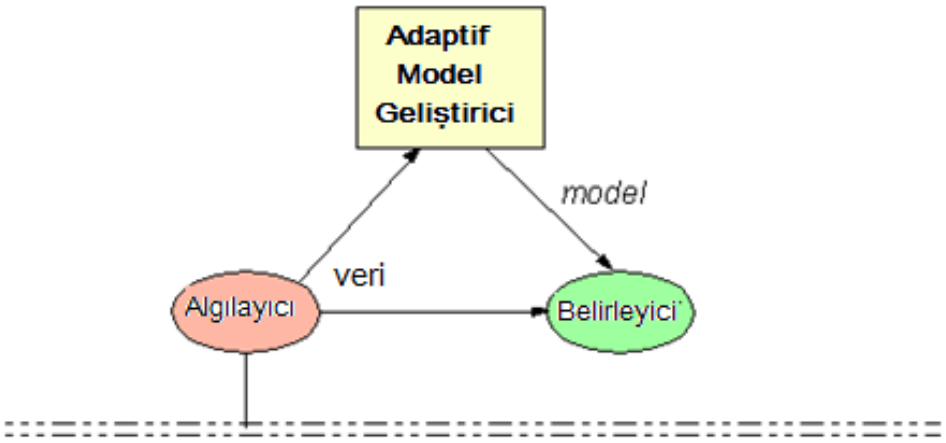
Eğitim aşamasının başında, küme modelin maksimum boyutu (toplam küme sayısı) ve yaprakların boyutu (her bir yapraktaki maksimum küme sayısı) sisteme girilmektedir. Girilen bir veri vektörü için modeldeki en yakın küme araması yapılmakta olup; eğer eşik değer kısıtı karşılanıyorsa, yeni veri noktası en yakın küme ile birleştirilmekte, aksi takdirde modele yeni bir küme eklenmektedir. Eğer model maksimum küme sayısına ulaşırsa, eşik değer artırılarak model yeniden oluşturulmakta ve yeni veri noktası modele eklenmektedir. Belirleme aşamasında, eğitimde olduğu gibi, indeks yapısı da kullanılarak modeldeki en yakın küme araması yapılmaktadır. Yeni veri noktası ile bulunan kümenin merkezi arasındaki uzaklık hesaplanmakta; eğer bu mesafe belirlenen sınırın altındaysa, yeni veri noktası normal; değilse anomali olarak kabul edilmektedir.

Algoritmanin gerçek-zaman özelliklerini test etmek için Veritabanında Bilgi Keşfi Kupası (KDD CUP99) veri seti kullanılmıştır. Belirleme oranı % 95, yanlış pozitif oranı % 2.8 olarak sonuçlanmıştır. Daha sonra, algoritma, gerçek bir telekom yönetim ağına benzemesi için tasarlanmış olan Safeguard test ağında da test edilmiştir.

[48] numaralı çalışmadaki “adaptif olasılıksal” model, veri toplama ve sızma belirleme modellerini oluşturma işlemlerini aynı zamanda yapmaktadır. Böylece ayrı bir öğrenme verisetine ihtiyaç duyulmamaktadır; çünkü veri seti oluşturma maliyeti çok yüksek olabilmektedir ve bu maliyet SBS geliştirmenin önünde önemli bir engel olarak durabilmektedir. Algoritma, belirleme modelleri geliştirirken normal veri içerisine karıştırılmış az miktarda sızma verisini tolere etmektedir. Tamamen otomatik olarak anomali belirleme modelleri geliştirmenin yanısıra, kullanıcıların kolaylıkla ve çabukça kötüye-kullanım belirleme için veri setleri ve modeller geliştirme ve bunları belirleyicilere dağıtma kabiliyetlerini önemli derecede geliştirmek için mekanizmalar da sunulmaktadır.

Adaptif model geliştirme çerçevesi dağıtık bir sistem olarak tasarlanmıştır. Model geliştirme işlemi, korunan sistemin kaynak tüketimini azaltmak için ayrı bir sistemde gerçeklenmektedir. Model geliştirme ve sızma belirleme birimleri, ölçümlerin

yapıldığı ağda sadece algılayıcılar kalacak şekilde ağ dışında başka bir ortamda çalışabilmektedir. SBS yapısında Şekil 2.27'de görüldüğü gibi algılayıcı, belirleyici ve adaptif model geliştirici olarak üç bileşen yer almaktadır. Algılayıcı, analiz için ve gerçekleşen sızmalara cevap verilmesi için uygun formattaki veriyi belirleyiciye, yeni belirleme modelleri öğrenilmesi için de adaptif model geliştiricisine gönderir. Geliştirilen yeni model, belirleyiciye verilir.



Şekil 2.27 : Adaptif Model Oluşturma yapısı [48].

Elamanların anomali olup olmadığının belirlenmesi için olasılık dağılımı kullanılmaktadır. Bu kapsamda, her iki durumun da benzerlik (likelihood) değerleri değerlendirilmektedir. Modelde, 1999 DARPA Intrusion Detection Evaluation (IDEVAL) verisetinin Basic Security Module (BSM) verilerinde yer alan, UNIX makinelerde yönetici haklarına sızmaya çalışan "user to root" saldırılarını içeren sistem çağrı verileri incelenmiştir. Daha çok eğitim verisi kullanıldıkça belirleme performansı artmaktadır.

[49] numaralı çalışmada, gürültülü veri üzerinde anomali belirleme yapan "Kuvvetli Destek Vektör Makineleri (Robust Support Vector Machines (RSVMs))" algoritması kullanılmaktadır. RSVM'lerde kullanılan destek vektörü sayısı standart SVM'lerdekinden daha az olduğu için test zamanı daha hızlıdır. Kullanılan yöntem, bir Solaris bilgisayardan toplanmış olan 1998 DARPA BSM verisetiyle değerlendirilmekte olup, standart SVM ve k-en yakın komşu sınıflandırıcısı (k-Nearest Neighbor classifier (kNN)) yöntemleriyle karşılaştırılmaktadır. Gürültülü veriseti oluşturmak için 28 adet sızmanın 16 tanesi normal veriymiş gibi verisetinde değiştirilmiş olup, %3 yanlış-pozitif ile beraber %100 doğru tespit sonucuna ulaşılmıştır.

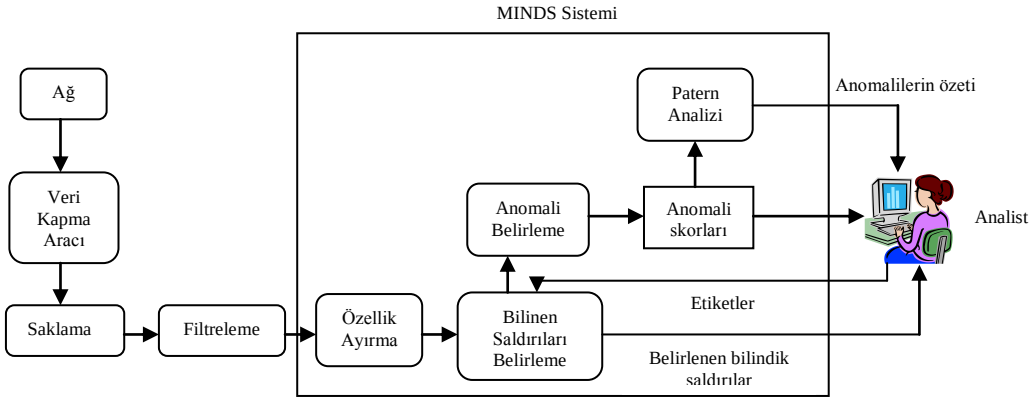
Support Vector Machines yönteminde, iki sınıf (pozitif-negatif) arasındaki ayırıcı bölgeyi genişleten bir düzlem oluşturulur. Bu ayırıcı bölgenin sınırında yer alan özellik vektörlerine "destek vektörleri", bu özelliği kullanan sınıflandırıcılara da "destek vektör makineleri" denilmektedir.

MINDS [50] anomali belirleme modülünde, her bir ağ bağlantısına bir anomali skoru verebilmek için "aykırı-değer belirleme (outlier detection)" algoritması kullanılmaktadır. Anomali skorları verildikten sonra, gerçek saldırı olup olmadığı kararı, sadece en büyük skora sahip bağlantıları inceleyen bir analist tarafından verilmektedir. Minnesota Üniversitesi ağ trafiğinden alınan ve sadece paket başlık bilgilerini içeren NetFlow verileri kullanılmıştır. Veri anomali belirleme modülüne girilmeden once, analistin analiz sırasında ilgi duymayacağı ağ trafiğini elemek için analist tarafında bir veri-filtreleme basamağı gerçeklenir. Ağ bağlantılarının etiketleri (normal/sızma) bulunmadığı için, belirleme oranı hakkında bir tahminde bulunulmamaktadır fakat, kullanılan anomali belirleme algoritması tarafından yüksek derecelendirilen bağlantıların tamamına yakını ağ güvenlik analistleri tarafından anomali olarak değerlendirilmiştir.

MINDS'daki ilk basamak, Şekil 2.28'de görüldüğü gibi veri madenciliği analizinde kullanılan özelliklerin çıkartılmasıdır. Temel özellikler, kaynak ve hedef IP adresleri, kaynak ve hedef portları, protokol, bayraklar, byte sayısı ve paket sayısından oluşmaktadır. Özellik oluşturma basamağından sonra, imzaları bulunan saldırılara denk gelen ağ bağlantılarını belirlemek ve ileriki analizler için ortadan kaldırmak için bilinen saldırıları belirleme modülü kullanılır. Sonrasında, veri, her bir ağ bağlantısına bir anomali skoru vermek için aykırı-değer (outlier) belirleme algoritması kullanan MINDS anomali belirleme modeline girilir. Bu algoritmada, veri noktalarına aykırılık derecelerine göre yerel aykırılık faktörü (local outlier factor (LOF)) verilmekte ve komşu bölgenin yoğunluğu hesaplanmaktadır. En sonunda, bir analist, en çok anomali gösteren bağlantılara bakarak gerçek saldırı olup olmadıkları kararını verir.

"Kendisi Organize olan Harita (Self Organizing Map(SOM))" [51] algoritması, çok boyutlu girdi uzayını, düzenli iki-boyutlu nöron dizileri haline getirmektedir. Haritadaki nöronlar, kod-kitabını (codebook) oluşturan n-boyutlu (girdi sayısı) referans vektörleriyle eşleştirilirler. Komşu nöronlar, komşuluk ilişkisi aracılığıyla birbirlerine bağlıdırlar. Kod-kitabındaki referans vektörleri girdi verisinin yüksek

olduğu alanlara kayarlar. En sonunda, girdi verisinin seyrek olduğu alanlarda çok az kod-kitabı vektörü kalır.



Şekil 2.28 : MINDS sistemi [50].

SOM algoritmasının öğrenme prosesi:

1. Girdi veri setinden rastgele bir vektör seçilir ve Euclid uzaklık ölçümüne göre kod-kitabı vektörlerine olan benzerliği (uzaklığı) hesaplanır.
2. En uygun birim (Best Matching Unit-BMU) bulunduktan sonra kod-kitabı vektörleri güncellenir. BMU ve komşuları, girdi vektörüne yakın olarak yerleştirilirler, çünkü girdi vektörlerinin çekim alanına girerler. Bu çekim alanının gücü, öğrenme derecesiyle kontrol edilir. Öğrenme ilerledikçe ve yeni girdi vektörleri haritaya eklendikçe, öğrenme derecesi yavaş yavaş sıfıra kadar azalır. Öğrenme derecesiyle beraber, komşuluk yarıçapı da küçülür.

Tasarlanan sistem, bir konakçı-temelli SBS olup uygulama ile ilgili bilgi verilmemiştir. [-1, 1] aralığında olacak şekilde normalize edilen özellikler, kullanıcının,

- Aktif olduğu saatler,
- Oturum açdığı konakçılar,
- Uzaktan eriştiği yabancı konakçılar,
- Kullandığı komut seti,
- MIB kullanımı,
- Hafıza kullanımı,

özelliklerini kapsamaktadır.

PAYL [52], paketleri yüklerinin uzunluğuna göre sınıflandırmakta, byte-sıklık-dağılımlarını değerlendiren istatistiksel fonksiyonları temel almaktadır (n-gram analizi). N-gram, yük ünitesindeki n adet bitişik bitin sıralamasıdır. Yük üzerinde n genişliğindeki bir kayan pencere geçer ve her bir n-gram'ın sayısı tutulur. Yeni verilerin profille olan benzerliklerini hesaplamak için Mahalanobis uzaklığı kullanılmaktadır. Belirleyici, bu ölçüleri bir eşik değerle karşılaştırır ve yeni girdinin mesafesi bu eşiği geçiyorsa bir uyarı verdirir. Modelleme yöntemi, tamamen denetemsizdir ve eğitim verisindeki gürültüye karşı toleranslıdır.

Yöntem, DARPA 1999 veriseti ve Columbia Üniversitesi Bilgisayar Bilimleri bölümü ağından toplanılan canlı veriseti ile test edilmiştir. DARPA 1999 verisetinde, TCP tarfiği için eğitilen en iyi model portların her birinin yanlış pozitif oranı % 1'in altında olacak şekilde, 97 saldırıdan 57'sini belirleyebilmiştir. 80'inci port için % 0.1 yanlış pozitif ile yaklaşık %100 doğruluk değeri elde edilmiştir. Aynı zamanda, denetimsiz öğrenme yöntemi oalrak kullanıldığında, CUCS verisetindeki Code Red II ve tampon taşması saldırılarını başarıyla belirlemiştir.

POSEIDON [53] (Payl Over Som for Intrusion DetectiON) çalışması da yük-temellidir ve iki-aşamalı yapısı vardır: ilk aşama yük verilerini sınıflandırmada kullanılan bir Self-Organizing Map (SOM) [54] içerirken, ikincisi yeniden düzenlenmiş bir PAYL sistemidir. PAYL çalışmasına yapılan değişiklik, her paketin SOM kullanılarak önceden-işlenmesidir ve yük uzunluğu yerine SOM tarafından verilen kazanan-nöron değerinin kullanılmasıdır. POSEIDON'da paketlerin yük verileri incelenmektedir, fakat izlenilen portların profillerini oluşturabilmek için, başlık verisi olarak sadece hedef adresi ve servis port numarası bilgileri kullanılmaktadır. DARPA 1999 verisetinin TCP trafiğinin bir bölümü üzerinde yapılan testlerde ortalama % 73 doğru pozitif, % 1 yanlış pozitif oranı elde edilmiştir.

PHAD [55], saldırıların çoğunluğu protokol uygulama hatalarını suiistimal ettikleri için, kullanıcı davranışlarından ziyade protokolleri modelleyen bir anomali belirleme algoritmasıdır. Sadece IP adreslerini ve port numaralarını değil, diğer (33 adet) paket başlığı alanlarını da inceleyen "olasılıksal" zaman-temelli bir anomali belirleme algoritması oluşturulmuştur. 8 bit'ten küçük alanlar (TCP bayrakları gibi) tek bir byte alanına gruplandırılmıştır. 4 byte'tan büyük olan alanlar (6 byte Ethernet adresleri gibi) yarıya bölünmüştür.

Gerçek-zamanlı trafiğin dinamik davranışını değerlendirebilmek için, belirleme modunda durağan-olmayan bir model kullanır. Bu modelde, eğer bir olay t saniye önce gerçekleşmişse, sonraki bir saniyede gerçekleşme olasılığı yaklaşık 1/t'dir. DARPA 1999 verisetindeki ulaşım katmanı ve aşağısındaki katmanlarda meydana gelen saldırıların çoğu tespit edilebilmektedir. DNS, HTTP veya SMTP gibi uygulama katmanı protokolleri incelenmediği için sunuculardaki saldırıları tespit edilememektedir. PHAD-C32, günde 10 yanlış alarm oranı (toplam 100) ile 201 saldırı örneğinin 67'sini belirleyebilmektedir.

[56] numaralı çalışmada, gelen veri paketlerinin kısa-zamanlı pencere analizlerinde tahmin edilen istatistiksel özellikleri temel alan "Radial-basis-function (RBF) sinir ağları yöntemi" ile DDoS saldırıları belirlenmektedir. Ağın verimliliği, 1-20 aralığında bir sayıdaki gizli nöronlar için ölçülmüştür. RBF'nin lineer olmayan bir fonksiyonu olarak, Gaus fonksiyonlarının karışımı kullanılmıştır. Gaus fonksiyonunun ortalaması ve varyansı k-ortalama kümeleme algoritmasıyla tahmin edilmiştir. Özellik tahmin modülü hızlı olduğu için, uygulamalarda küçük zaman dilimleri tercih edilmektedir. 6 sn'lik veri penceresi içerisinde tahmin edilen üç adet istatistiksel özelliği kullanarak simule bir genel ağdaki bilinen DDoS saldırılarının %98'den fazlası tespit edilmiştir.

[57] numaralı çalışmada, normal davranış istatistikleri üzerine eğitilen modeller yerine, TCP/IP protokollerinin ayrıntılı modelleri oluşturulmaktadır. İstatistiksl anomali belirlemenin ağ trafik istatistiklerinin normal modelini yaratma konusunda sıkıntıları vardır. Protokoller iyi tanımlandığı için protokol anomali belirleme daha kolaydır ve normal model daha kesin bir doğrulukla yaratılabilmektedir. Bütün bağlantı yönlü protokollerin durumları (state) vardır: belirli zamanlarda belirli olaylar meydana gelir. Buna dayanarak protokol anomali belirleyicilerin çoğu, "sonlu durum otomatları (state machines)" şeklinde geliştirilirler. Her bir durum, istemciden cevap bekleyen sunucu gibi, bağlantının bir bölümüne denk gelmektedir.

Kötü niyetli saldırı imza uzayı inanılmaz bir oranla büyümektedir. Bunun gibi, saldırı imza veritabanları da saldırıları etkin olarak belirleyebilmek için sık sık güncellenmelidir. Buna karşılık, yeni protokoller ve varolan protokollerin uzantıları daha yavaş bir oranla geliştirilmektedir. Ağ protokolleri uzayı, iyi tanımlıdır ve yavaş yavaş değişir. Protokol anomali belirleyiciler yeni saldırıların çoğunu

güncellenmeden belirleyebilir çünkü yeni saldırılar protokol özelliklerinden sapmaktadır.

[58] numaralı çalışmada tasarlanan anomali belirleyicisi, ağ trafiği hakkında bilgi toplayan, analiz eden ve ağdaki anomalileri belirleyen bir Snort ön-işlemcisidir. Ağ trafiğinin 25 parametresi kaydedilmektedir. Veri toplama modunda bir süre çalıştıktan sonra her parametrenin ortalama ve standart sapma değerlerini içeren "ağ profili" oluşturulmaktadır. Anomali belirleme modunda, bu "istatistiksel" değerlerden olan sapmalar değerlendirilmektedir.

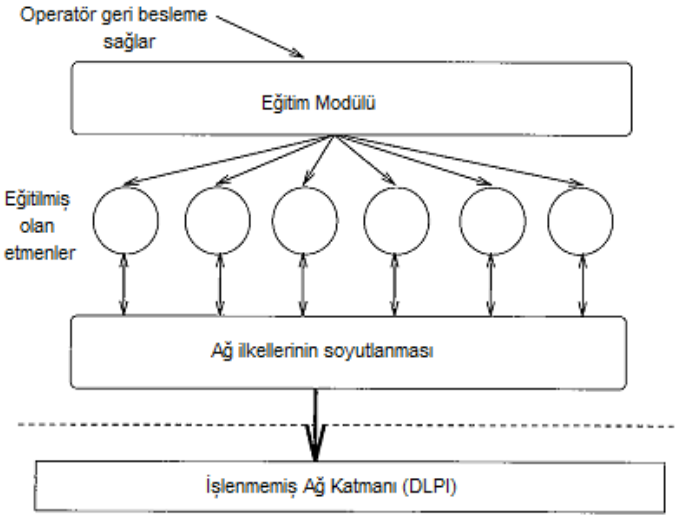
Yenilikçi SBS Bağışıklık Sistemi Modelinde [19] "kendine-ait" sınıfında dört farklı gen grubu vardır. Bunlar, kullanıcı genler, sistem genleri, işlem genleri ve ağ genleridir. "Kendine-ait" olanların tanımları test verilerinden ve önceki deneyimlerinden elde edilir. Bu dört gen grubu, negatif seleksiyon algoritmasında "kendine-ait" kümesini oluşturmaktadır. Her gen bankası, kendi gen bankasını oluşturabilir ve geliştirebilir. Her sistem, ağa ajanlarını bağımsız olarak gönderir. Bu ajanların akışkanlığı ve adaptifliği vardır. Düğümler arasında serbestçe dolaşırlar ve evrim mekanizması sayesinde yüksek adaptif ajanlar oluştururlar.

Belirleme süreci şu şekilde işlemektedir: 1) Dört gen grubu bağımsız olarak negatif seleksiyon algoritması yardımıyla olgun etmenleri oluşturur ve ağa gönderirler. 2) Eğer etmenler sızmalarla eşleşirlerse, kendilerini kopyalar ve diğer bölgelere bu kopyaları gönderirler. Aynı zamanda, yaratıldığı gen bankasına da bir kopyasını gönderir. Eğer hiçbir sızma ile eşleşmez ise bir süre sonra yok olurlar. 3) Model, sistem parametreleri belli bir eşiği aşınca yöneticiyi uyarır.

Genetik Algoritma'nın global optimize etme ve Tavlama Benzetmesi'nin yerel optimizasyondan kaçınma özelliklerini birleştirerek, gen bankasının oluşturulması ve geliştirilmesini gerçekleştirildiği anlatılmakta olup uygulamadan behsedilmemektedir.

Bir bilgisayar sisteminin korunmasını konu alan [7] numaralı çalışmadaki özerk etmenler, denetim bilgilerini izlemek için sızma profillerini öğrenmek zorundadırlar. Öğrenme için Genetik Programlama (GP) seçilmiştir. Öğrenme modeli, etmenlere bir senaryonun sunulduğu ve performanslarına dayalı olarak değerlendirildikleri geri besleme yöntemini kullanmaktadır. Eğitim sonuçları biriktirilir ve sonra en iyi etmenler SBS'e kullanılmak için yerleştirilirler.

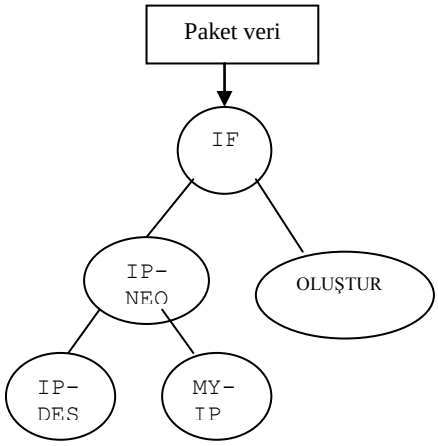
GP sisteminde, program popülasyonları spesifik bir problemi çözmek için evrimleştirilirler. Problemin genelde tek bir çözümü yoktur veya çözümün hesaplaması çok pahalıdır. Şekil 2.29'da etmenlerin nasıl çalıştıkları görülebilmektedir. En altta işlenmemiş ağ arayüzünün kendisi bulunur. Bu prototip uygulamada, Sun DLPI arayüzü kullanılmıştır.



Şekil 2.29 : Sistemin çalışması [7].

Etmen kodu, operatörler (aritmetik, mantıksal ve şartlı ifade) ve metriklerin değerlerini alan ilkel kümelerinden oluşur. Bu kümeler, çözüm programları için inceleme (parse) ağaçları oluşturmak amacıyla değişik yollarla birleştirilebilirler. Sonrasında, bu programlar bir eğitim senaryoları kümesine karşı değerlendirilirler; her bir potansiyel etmene bir uygunluk skoru atanır. Bu skor, etmenin eğitim senaryosunu ne kadar iyi sınıflandırabildiğine dayalıdır. Şekil 2.30'da basit bir inceleme ağacı yer almaktadır. Bu ağaç, aşağıdaki sahte-kod bloğuna denk gelmektedir:

```
For- each- packet- do
  İf(ip- destination- address- of- packet
        is- not- equal –to- my-  ip- address)
     Then- generate- a- suspicion- broadcast
  Endif
Endfor
```



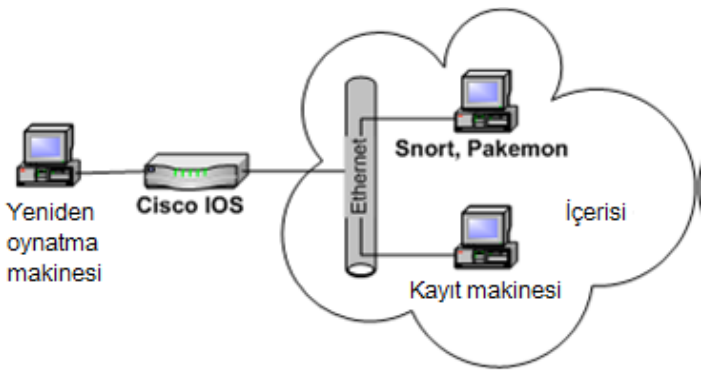
Şekil 2.30 : Basit İnceleme Ağacı [7].

[59] numaralı melez-SBS çalışmasında, PHAD ve NETAD [67] çalışmaları deneysel olarak geliştirilmiş ve Snort [13]'a bir ön-işlemci olarak eklenmiştir. NETAD da PHAD gibi paketleri modellemektedir ve iki aşamada çalışmaktadır: ilk aşama, oturumların başlangıcını belirlemek için gelen istemci oturumlarını filtrelemektir. İkinci aşama da modelleme aşamasıdır. Filtreleme aşamasında trafiğin %98-99'u elenmektedir. Bu eleme modelleme aşaması öncesi trafiği sadeleştirmektedir. Saldırıların kanıtlarının bulunduğu trafik verisi, modelleme aşamasına gönderilmektedir. İkinci aşama dokuz çeşit paketi modellemektedir. Toplamda 9 x 48 = 432 kural mevcuttur. Bu yaklaşım, daha az yanlış alarm verdiren kuralların ağırlığını artırmaktadır.

Snort'u PHAD ve NETAD ile birleştirmek için Snort'un önişlemci yapısı kullanılmıştır. Önişlemciler, Snort'un ana belirleme makinesine ulaşmadan önce paketleri değiştirme, yok sayma ve uyarılar verdirme kabiliyetlerine sahip makinelerdir. Snort + PHAD + NETAD birleşimi DARPA 1999 verisetindeki 201 saldırıdan 146'sını belirleyebilmiştir.

[60] numaralı çalışmada, iki açık-kaynak ağ sızma belirleme sistemi olan Snort ve Pakemon [61] saldırıların belirlenmesini artırmak için Cisco IOS Firewall'un [62] sızma belirleme özellikleriyle Şekil 2.31'de gösterildiği gibi birleştirilmiştir.

Cisco IOS Firewall'un, ateşduvarı özellikleri yanında, bilinen saldırıları ve kötüye-kullanım girişimlerini belirleyebilmek için 59 adet statik imzası vardır. Ateşduvarı üzerindeki SBS işlemi, bu 59 imzayı kullanarak paket başlıklarını sızma belirleme için incelemektedir. Pakemon, parçalanma, düzensizlik, tekrar, üst-üste binme, ekleme ve IP veya TCP katmanda senkronizasyonsuzluk gibi kaçış yöntemlerini belirlemeyi amaçlayan bir açık kaynak deneysel SBS'dir.



Şekil 2.31 : Sistemin ağ diyagramı [60].

Önceden tcpdump programı kullanılarak toplanılan paketleri canlı ağda tekrar hareketlendirmek için, SourceForge.net tarafından sağlanan TCPReplay aracı kullanılmıştır. Bu üç aracı birleştirince, DARPA 1999 verisetinde yaklaşık % 56 belirleme oranı elde edilmektedir.

K-Means+ID3 [63], k-ortalama kümeleme ve ID3 karar ağacı öğrenmesi adlı iki makine öğrenme algoritmasının birleştirilmesiyle geliştirilmiştir. K-ortalama kümeleme yöntemi, öncelikle eğitim örneklerini Euclid mesafesi benzerliğini kullanarak, k adet kümeye ayırır. Her bir kümede, normal ve anomali örneklerinin yoğunluk bölgelerini temsil edecek şekilde, ID3 karar ağacı oluşturulur. Her bir kümedeki karar ağacı, küme içerisindeki alt-grupları öğrenerek karar sınırlarını netleştirir. K-ortalama ve ID3 yöntemlerinin kararları, sınıflandırma hakkındaki son kararı elde etmek için iki kural kullanılarak birleştirilirler: 1) En yakın komşu kuralı ve 2) En yakın anlaşma kuralı.

Eğitim veri setlerindeki örneklerin sayısı, %70'i normal ve geri kalanı anomali olacak şekilde en fazla 5000 örnek ile sınırlandırılmıştır. DARPA 1998 verisetinde % 76 doğru pozitif oranı ve % 5 yanlış pozitif oranı, DARPA 1999 verisetinde % 99.12 belirleme doğruluğu elde edilmiştir.

jREMISA [5] çalışmasında, en iyi sınıflandırma uygunluk derecesine ve çok-amaçlı hiper-hacim büyüklüğe sahip iyi belirleyiciler elde etmek için Yapay Bağışıklık Sistemi [42] bir Çok-amaçlı Evrimsel Algoritma [64] ile beraber kullanılmıştır. Ağ trafiği, bir veriseti kullanılarak eğitilmiş olan antijen belirleyicilerin yardımıyla kendinden ve kendinden-olmayan olarak sınıflandırılmıştır. Çözüm yaklaşımı, mevcut iki adet AIS-kullanan algoritmayı geliştiren yeni bir algoritma sunmaktadır. Bu iki çalışma: string eşlemeye göre lokal optimumdan daha iyi kaçma kabiliyetinden dolayı Edge, Lamont and Raines'in retrovirus algoritması (REALGO) [65] ve AIS'i çok-hedefli EA bağlamında gerçeklediği için Coello and Cortés'in çok-hedefli bağışıklık sistemi algoritmasıdır (MISA) [66]. jREMISA hakkında detaylı bilgi ilerleyen bölümlerde verilmektedir.

Çizelge 2.4'de Anomali-temelli SBS çalışmalarında kullanılan öğrenme yöntemleri, testlerde kullanılan verisetleri ve varsa sonuçlarının ele alındığı karşılaştırma tablosu yer almaktadır.

Çizelge 2.4 : Anomali-temelli SBS çalışmalarının karşılaştırılması.

S/N	Çalışmalar	Kullanılan yöntemler	Kullanılan Veriseti	Sonuçlar
1	[8]	Kümeleme (clustering)	KDD CUP99	TP %95, FP %2.8
2	[48]	Adaptif olasılıksal	MIT-DARPA99 BSM verilerindeki "user to root" saldırıları	TP %100, FP %0.02
3	[49]	Kuvvetli Destek Vektör Makineleri (RSVM)	MIT-DARPA98 BSM uyarlaması	TP %100, FP %3
4	[50]	Aykırı-değer belirleme (outlier detection)	Minnesota Üniversitesi NetFlow verileri	-
5	[51]	Kendisi Organize olan Harita (SOM)	-	-
6	[52]	n-gram azalizi	MIT-DARPA99, sadece TCP trafiği	TP %59, FP <%1
7	[53]	SOM + n-gram azalizi	MIT-DARPA99, sadece TCP trafiği	TP %73, FP %1
8	[55]	Zaman temelli olasılıksal	MIT-DARPA99	TP %33, FP 10 adet/gün
9	[56]	Radial-basis-function (RBF) sinir ağları yöntemi	Simule bir ağ trafiğindeki bilinen DDoS saldırıları	TP %98
10	[57]	Protokol (TCP) anomali belirleme	-	-
11	[58]	İstatistiksel SNORT önişlemcisi	-	-
12	[19]	Yapay bağışıklık sistemi	-	-
13	[7]	Genetik programlama	-	-
14	[59]	SNORT + Zaman temelli olasılıksal (PHAD + NETAD)	MIT-DARPA99	TP %73
15	[60]	SNORT + Pakemon + Cisco IOS Firewall	MIT-DARPA99	TP %56
16	[63]	k-ortalama kümeleme ve ID3 karar ağacı öğrenmesi	MIT-DARPA99, sadece 5000 örnek paket	TP %99
17	[5]	Yapay Bağışıklık Sistemi + Çok-amaçlı Evrimsel Algoritma	MIT-DARPA99, eğitim aşaması	TP %99, FP %15

3. GEZGİN ETMENLER KULLANAN DAĞITIK İMZA-TEMELLİ SIZMA BELİRLEME

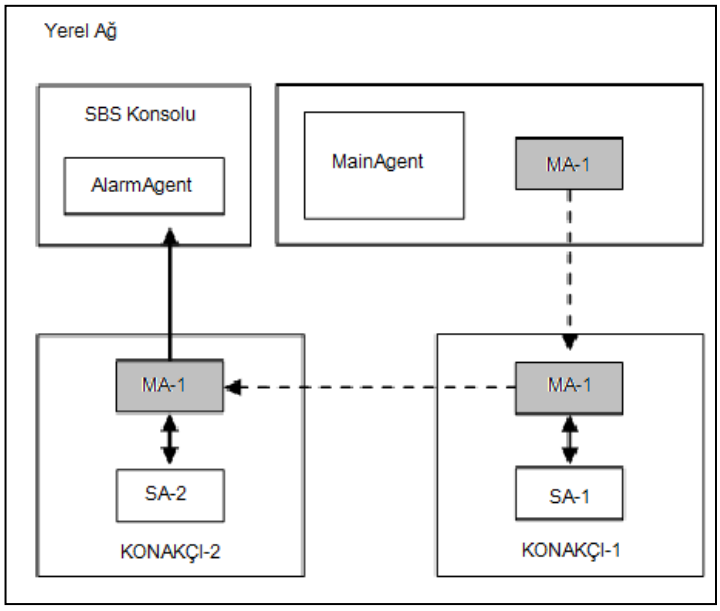
Tezin bu bölümünde, MIT DARPA LLS_DDOS 1.0 verisetinde yer alan Dağıtık Hizmetin Reddedilmesi saldırılarını, saldırı yapılırken aracı olarak kullanılan ara ağda tespit etmek ve saldırı tamamen gerçekleşmeden önce gerekli uyarıları verebilmek amaçlı altı farklı dağıtık sızma belirleme yöntemi oluşturulmuş ve gerekli testler yapılarak alınan sonuçlar karşılaştırmalı olarak yorumlanmıştır. Bu yöntemlerden ilk ikisinin benzerleri başka çalışmalarda da denenmiş olup diğer dördü ise ilk defa oluşturulmuştur. Bütün yöntemlerde, imza-temelli bir ağ sızma belirleme sistemi olan Snort tarafından yaratılan alarm verileri kullanılmıştır. Birinci yöntem hariç yöntemlerin hepsinde, veri analiz hesaplamalarını sızma verilerinin bulunduğu yerlere kaydırarak ağ trafiğini rahatlatmak amacıyla, Jade platformunda çalışan gezgin etmenler kullanılmıştır. En sonunda, yöntemler, güvenilirlik, ağdaki yoğunluk ve ortalama sızma belirleme süresi değerlerine göre karşılaştırılmışlardır.

Konakçılardaki alarmları oluşturması için, akademik araştırma projelerinde yaygın olarak kullanılan imza-temelli bir SBS olan Snort seçilmiştir. Snort, PHAD [55] and NETAD [67] SBS'lerinin deneysel olarak geliştirildiği ve Snort'a önişlemci olarak eklendiği [59] numaralı çalışmada, tek başına, 1999 DARPA verisetindeki 201 saldırı örneğinin 27 tanesini % 14 doğru belirleme oranı ile belirleyebilmiştir. [60] numaralı çalışmada, Snort ve bir diğer açık-kaynak ağ sızma belirleme sistemi olan Pakemon [61], daha çok saldırı belirlenebilmesi amacıyla Cisco IOS Firewall'un [62] sızma belirleme özellikleri ile birleştirilmiştir. Bu birleşme sonrasıda Snort, yukarıda belirtilen verisetinin dördüncü haftası ile test edildiğinde % 44 doğru belirleme oranı sağlamıştır. 1999 DARPA SBS veriseti deneylerindeki düşük belirleme oranlarına rağmen, bu tez çalışmasında kullanılan 2000 DARPA LLS_DDOS 1.0 verisetindeki DDoS saldırısının ilk üç aşamasını başarıyla belirleyebilmiştir.

3.1 Sistem Tasarımı

Altı yöntem de Şekil 3.1'de gösterilen DIDMA [3] yapısı üzerinde bazı değişiklikler yapılarak oluşturulmuştur. Temel farklılıklar: Static Agent'lardaki SBS olarak Snort kullanılması, gezgin etmen platformu olarak Jade kullanılması, daha detaylı bir DDoS veriseti kullanılması, sızma belirleme aşamalarının daha kapsamlı işlenmesi ve Mobil Agent'ların farklı yaratılma şekilleridir. Birincisi dışındaki SBS yöntemlerinin tamamında gezgin etmenler kullanılmıştır. Bu çalışmada sunulan SBS yöntemlerinin bileşenleri:

- MainAgent,
- Static Agent'lar,
- Mobile Agent'lar,
- AlarmAgent'dır.



Şekil 3.1 : Sistemin genel yapısı.

Yerel ağdaki her bir konakçıda, o konakçıyı sürekli olarak izleyen ve bulunduğu konakçıda şüpheli bir olay belirlediğinde MainAgent'ı haberdar eden bir Static Agent yer almaktadır. MainAgent'ı barındıran konakçının bir şekilde kullanılamaz hale gelmesi durumunda, güvenli bir alanda kurulu olduğu varsayılan AlarmAgent'ın, sızma tehlikesi hakkında güvenlik yöneticisini uyarmak amacıyla SBS konsoluna mesaj göndermeye devam edebilmesi için, MainAgent ve Alarm Agent ağda farklı yerlere yerleştirilmiştir. MainAgent, şüpheli olaylar hakkında bilgi

toplamak amacıyla ağda dolaşan ve en sonunda gerçekten bir sızma olup olmadığı kararını veren saldırıya-özel Mobile Agent'lar yaratmaktadır. Bu gezgin etmenlerin birbirleriyle ve diğer etmenlerle haberleşirken güvenli olarak asıllama yaptıkları varsayılmaktadır. Bu belirleme süreci, ileride daha detaylı anlatılacağı gibi, önerilen yöntemlere göre farklılıklar arzetmektedir.

3.1.1 Yöntem_1

Hâlihazırda en çok kullanılan ve eleştirilen bu merkezi yöntemde, MainAgent tek başına bütün işleri yapmaktadır. Bu yöntemde, MobileAgent ve AlarmAgent kullanılmaz; SBS konsolu MainAgent ile aynı yerdedir. Konakçılardaki Static Agent'lar merkezdeki MainAgent'a şüpheli olay hakkındaki tüm verilerin yer aldığı sızma şüphesi mesajları gönderir. Aynı tipte olup da farklı konakçılardan gelen sızma mesajlarının sayısı kısıtlı bir süre zarfında (bir saat) birden fazla (iki) olursa, MainAgent, SBS konsola ilgili dağıtık sızma hakkında bilgi verir ve log dosyasına kaydeder.

Bütün işlerin MainAgent tarafından merkezi olarak yapılması sistemin hızlı çalışmasına neden olacaktır. Fakat şüpheli olaylar hakkında veri toplama, analiz etme ve dağıtık sızma kararı verme sorumluluklarının hepsi MainAgent'ta olduğu için merkeze gönderilen sızma şüphesi mesajlarında fazlaca veri olmak zorundadır. Yapılan testlerde imza-temelli belirleme sistemi için yalnızca saldırıcının IP adresi kullanılması yeterli görülmüştür. Fakat, anomali-temelli sistemlerde, bu sızma şüphesi mesajı içerisinde saldırıdan etkilenen sistem dosyaları, registry kayıtları, ağ log dosyaları ve daha fazla veri bulunmak durumunda olabilir. Bu durum, ağda veri akışı yoğunluğuna sebep olacaktır. Ayrıca, MainAgent'ın kullanılamaz hale gelmesi veya getirilmesi durumunda sistem çalışamaz hale gelecektir.

3.1.2 Yöntem_2

Bu ve sonraki yöntemlerde Mobile Agent'lar ve AlarmAgent MainAgent'a yardımcı olmaktadır. Başta anlatıldığı gibi, güvenlik avantajından dolayı, SBS Konsolu ve MainAgent ayrı yerlerde bulunmaktadırlar. Belirli bir süre içerisinde, farklı konakçılardan aynı ip numarasına sahip saldırıcıyı içeren toplam iki adet mesaj gelmesi durumunda, MainAgent, ilgili tipteki sızma kayıtlarını yerinde incelemek amacıyla bir MobileAgent yaratır ve o iki mesajı gönderen konakçılara gönderir. MobileAgent gittiği konakçıdan gerekli bilgileri (saldırıcıının ip numarası) alır ve

karşılaştırır. En sonunda aynı kaynaktan yapılan dağıtık bir sızma olup olmadığına karar verir ve kendisini yok eder. Eğer sızma varsa AlarmAgent'ı mesajla bilgilendirir. AlarmAgent da sızma bilgilerini konsolda görüntüler ve log dosyasına kaydeder.

Kullanılan yapı, örnek alınan DIDMA [3] yapısına benzemektedir. DIDMA çalışmasında, Static Agent'lar kendi konakçılarından kaynaklanan SYN ve UDP taşkınlarını Mobile Agent Yaratıcısı olan merkez üniteye bildirirler. Bu merkez ünite tarafından yaratılan ilgili saldırı tipindeki Mobile Agent, saldırı şüphesi mesajı gönderen konakçılara gider ve saldırı yapılan bilgisayarın IP adresinin aynı olup olmadığını kontrol ederek dağıtık saldırı kararını verir. Bu tez çalışmasında kullanılan DDoS veriseti ve sızma belirleme aşamaları DIDMA çalışmasındakilerden daha detaylı ve daha etkindir. Saldırı, ilk aşamalarında tespit edilerek güvenlik yöneticisine saldırının tamamı gerçekleşmeden durdurabilme imkânı verilmektedir. DIDMA'daki Voyager platformu yerine burada, Jade etmen platformu kullanılmıştır. Bir başka önemli fark da Static Agent'ların sızma şüphesi mesajları oluşturulurken Snort'dan yardım alması durumudur.

Bu yöntemde, şüpheli olaylar hakkındaki veriler Yöntem_1'de olduğu gibi merkeze taşınmaz; Mobile Agent'lar ilgili konakçıları ziyaret ederek olayları yerinde inceler, gerekli analizleri yapar ve hatta sistemde bir dağıtık sızma olup olmadığı kararını kendileri verirler. Böylece ağdaki yük yoğunluğu artmamış olacak ve MobileAgent yaratıldıktan sonra MainAgent bir şekilde kullanılamaz hale gelirse bile sistem doğru olarak çalışacaktır.

3.1.3 Yöntem_3

Yöntem_2'den farklı olarak; MainAgent, MobileAgent yaratmak için gelen mesaj sayısının iki olmasını beklemez, belirli bir süre içerisinde farklı bir konakçıdan farklı bir sızma şüphesi mesajı gelir gelmez ilgili tipte MobileAgent yaratır. MobileAgent, ilk olarak kendisinin yaratılmasına neden olan mesajı gönderen konakçıya gider ve ilgili kayıtlardan saldırıcıın ip numarasını öğrenir. Sonra rastgele olarak platformdaki diğer konakçıları dolaşmaya başlar. Bir turu tamamlayana kadar aynı konakçıya tekrar uğramaz ve diğer turlarda da aynı rastgele sırayı takip eder. Konakçılardaki kayıtları inceleyerek, belirli bir süre içerisinde aynı ip numaralı saldırıcıdan aynı tipte bir saldırı kaydını tespit ederse diğer konakçıları dolaşmaya gerek kalmadan, ilgili

saldırıcı tarafından yapılan bir dağıtık saldırı olduğu kararına varır ve bu durumu mesajla AlarmAgent'a iletir. AlarmAgent da sızma bilgilerini konsolda görüntüler ve log dosyasına kaydeder. Aksi takdirde, Mobile Agent ağdaki bütün konakçıları dolaşacak ve belirli bir süre sonra kendini sonlandıracaktır.

Yöntem_2'deki MobileAgent yaratılması için beklenilen ikinci mesaj bir şekilde MainAgent'a ulaşamazsa veya gelen ilk mesajdan sonra MainAgent kullanılamaz hale getirilirse bile, sistem bu yöntemde doğru olarak çalışacaktır. En kötü durumda Mobile Agent'ların bilgi toplamak amacıyla ağdaki bütün konakçıları dolaşması veya rastgele dolaşım sırasında dağıtık sızmaya maruz kalan diğer konakçının son sıralarda yer alması durumunda sistemin çalışması önceki yöntemlere göre daha uzun sürecektir.

3.1.4 Yöntem_4

Bu yöntemde, Yöntem_3'den farklı olarak; Mobile Agent'lar MainAgent tarafından değil de herbir konakçıda yer alan Static Agent'lar tarafından yaratılır. Gereksiz tekrarın önüne geçebilmek için, konakçıdaki Static Agent şüpheli bir olay belirlediği zaman öncelikle MainAgent ile haberleşerek, aynı zaman diliminde aynı tip başka bir MobileAgent'ın yaratılıp yaratılmadığı kontrolünü yapar. Eğer "hayır" cevabı alınırsa veya bu haberleşme belirli bir süre zarfında gerçekleşmezse, Static Agent ilgili Mobile Agent'ı yaratır ve platforma gönderir. Mobile Agent'ın platformdaki yolculuğu Yöntem_3'de olduğu gibi rastgeledir.

Bu yarı-dağıtık yöntemde, eğer MainAgent herhangi bir zamanda ve herhangi bir şekilde kullanılamaz hale getirilirse veya gelirse bile sistem doğru olarak çalışacaktır. Ancak tabii ki, Yöntem_3'de belirtilen en kötü durumda uzun zaman harcanması dezavantajı bu yöntemde de mevcuttur.

3.1.5 Yöntem_5

Yöntem_4'den farklı olarak, sızma şüphesi belirleyen konakçıdaki Static Agent, MainAgent ile hiç koordine kurmaksızın (diğer konakçılarla zaten koordine kurulmamaktadır) ilgili sızmayı araştırmak amacıyla bir Mobile Agent (MA) başlatır. Bu yöntemde, MainAgent hiç kullanılmamış olduğu için sistem tamamen dağıtık hale gelmiştir. Herhangi bir sızma şüphesi barındıran konakçılar, merkezle ve birbirleriyle koordinesiz olarak sisteme çok sayıda MA (aynı işi yapma olasılıkları

var) gönderirler. MA, Yöntem_4'de olduğu gibi bütün diğer konakçıları rastgele dolaşarak gerekli sorgulamayı yapar. Aynı kriterlere sahip sızma barındıran bir konakçı bulursa taramasını durdurur, ağdaki aynı işi yapan diğer MA'ları, konakçılara yayımladığı 'broadcasting' mesajı vasıtasıyla sonlandırır ve AlarmAgent'a bilgi mesajı gönderdikten sonra kendisini yok eder. Aksi takdirde, bütün konakçıları sorgulayarak dolaşır ve en sonunda kendisini sonlandırır.

Yöntem_4'de MA yaratılırken MainAgent ile yapılan haberleşme, bu yöntemde, ağdaki MA yükünü artırma pahasına iptal edilmiş olup; koordinesiz ve anında MA yaratılması yöntemiyle dağıtık sızmanın daha erken belirlenmesi/önlenmesi ve sistemin merkezi bir koordinatör olmaksızın tamamen dağıtık olması hedeflenmiştir.

3.1.6 Yöntem_6

Bu yöntemde, Yöntem_2'nin kısa ortalama belirleme süresi avantajıyla Yöntem_5'in tamamen dağıtık olan yapısı birleştirilerek güvenilirliği ve hızı yüksek birleşik bir sistem yaratılmıştır. Sistem, normal zamanda mod-1 (Yöntem_2)'de çalışırken; merkezi ünitenin kullanılamaz hale gelmesi durumunda otomatikman mod-2 (Yöntem_5)'ye geçiş yaparak sağlıklı çalışmasına devam edecektir.

Mod-1'de, konakçılardaki Static Agent'lar merkezdeki MainAgent'a sızma şüphesi mesajları gönderirler. MainAgent da gelen her mesaja karşılık olarak, çalışır durumda olduğu bilgisini içeren bir cevap mesajı gönderir. Aynı tipte olup da farklı konakçılardan gelen sızma mesajlarının sayısı kısıtlı bir süre zarfında (1 saat) birden fazla (iki) olursa, ilgili tipteki sızma kayıtlarını yerinde incelemek ve gereken kararı almak amacıyla bir MobileAgent yaratılır ve o mesajları gönderen konakçılara gönderilir.

Eğer ki, MainAgent'ın göndermesi gereken cevap mesajı 2 sn içerisinde ilgili Static Agent'a ulaşmaz ise, sistem otomatikman mod-2'ye geçer ve yukarda anlatıldığı gibi Yöntem_5 yapısında çalışmasına devam eder; çünkü artık MainAgent çalışamaz duruma gelmiştir ve sistem tamamen dağıtık olarak çalışmak zorundadır. Aynı zamanda, diğer bütün Static Agent'lara mod-2'ye geçmeleri gerektiği mesajı gönderilir. Uyarı mesajını alan Static Agent, eğer henüz kendisi otomatikman mod-2'ye geçmemişse, gelen bu mesajı kabul eder ve mod-2'ye geçer.

MainAgent, bir dağıtık sızmaya ait şüphe mesajının birincisini aldıktan sonra kullanılamaz duruma gelmiş ve aynı sızmaya ait ikinci mesajı gönderen konakçı gereken cevap mesajını alamadığı için mod-2'ye geçmiş ise, bu dağıtık sızmanın belirlenmesi işlemi yarım bırakılmayacaktır. Çünkü, mod-2'ye geçmiş olan konakçıdaki Static Agent tarafından yaratılacak olan Mobile Agent gerekirse bütün ağı dolaşarak ilgili sızmayı tespit edecektir.

3.2 Uygulama ve Deneyler

Bütün testler, Windows XP SP2 işletim sistemini kullanan, Pentium M 1.6 GHz, 736 MB RAM özelliklerindeki bir dizüstü bilgisayar üzerinde gerçekleştirilmiştir. Veri setinde aktif olarak yer alan 20 konakçı üzerinde çalışacak olan Static Agent'lar, MainAgent ve AlarmAgent ayrı birer DOS Command Prompt'ta yaratılan Jade container'larında çalıştırılarak ağ ortamı simule edilmiştir. Simule ağda, başka gerçek bilgisayarlar kullanılmadığı için test sonuçlarına ağ trafiği etki etmemiştir. Konakçılar birden fazla bilgisayarda çalıştırıldığında da sistem aynı sonuçları vermektedir.

3.2.1 Test tasarımları

Bir sızmanın dağıtık özelliğine sahip olması için aynı kaynak veya beraber çalışan birden fazla kaynak tarafından birden fazla konakçıya sınırlı bir zaman diliminde yapılmış olması gerekmektedir. Bu yüzden, dağıtık saldırı durumunu incelemek için yukarda belirtilen şartları içeren iki adet sızma şüphesi mesajı yeterli görülmüştür. Dağıtık saldırının gerçekleşeceği süre ortalama bir zaman dilimi olarak 1 saat seçilmiştir. Bu tür saldırıların gerçekleştirilmesi birkaç saniye sürebileceği gibi bir gün de sürebilmektedir.

Test ölçümlerinin gerçek zamanlı ağ trafiğinde yapılması için ilgili veriseti ağ trafiğinin yeniden canlandırılması akla gelen ilk çözüm olmakla beraber gerçeklenmesi zor bir seçenektir. Onun yerine, testlerin başlangıç zamanlarının veriseti trafiğinin başlangıç zamanıyla eşleştirildiği ve bütün sabit ve gezgin etmenlerin konakçılardaki saldırı kayıtlarını incelerken saldırıların gerçekleşme zamanlarını dikkate aldıkları bir yöntem gerçeklenmiştir. Etmenler, ilgili kayıtlarda tarama yaparken kendi saatleri ile kayıtlı saldırının gerçekleşme zamanını karşılaştırmakta, eğer henüz saldırı gerçekleşmemiş görünüyorsa, o ve sonraki saldırı

kayıtlarına hiçbir işlem yapmadan işlerine devam etmektedirler. Böylece, saldırılar önceden değil, gerçek oluşma zamanlarında tespit edilebilmekte ve ilgili birimlere bildirilebilmektedir. MA'ların ağda bir saat boyunca dolaşmaları anlamlı hale gelmiştir. Veriseti, yaklaşık 3 saatlik bir süreyi kapsamasına rağmen DDoS saldırısında ilgilenilen aşamalar 1 saatten daha az bir sürede gerçekleştiği için, testler, bu aşamaların olduğu bir saatlik zaman dilimi boyunca koşturulmuştur.

Yöntem_6'daki 20 koşturmanın 2 tanesinde (%10) MainAgent çalışamaz duruma getirilerek sistemin mod-2'de çalışması sağlanmıştır. Aynı istisnai durum Yöntem_4 için de geçerlidir. Böylece, Yöntem_6 ve Yöntem_4'ün çalışma zamanının %10'unda tam-dağıtık modda çalıştığı zamanlardaki belirleme süresi sonuçları elde edilmiştir.

Örnek olarak 2'nci yöntemin çalışma yapısı aşağıda açıklanmıştır:

- MainAgent, AlarmAgent ve 20 farklı konakçıda yer alan Static Agent'lar sırasıyla başlatılır.
- MainAgent, Static Agent'lardan sızma şüphesi mesajı; AlarmAgent da Mobile Agent'lardan sızma alarm mesajı beklemeye başlar.
- Static Agent'lar, her iki saniyede bir, ait oldukları konakçı'nın Snort tarafından oluşturulmuş olan "alarm.ids" dosyasına yeni kayıt olup olmadığını kontrol ederler. Eğer yeni bir sızma alarmı kaydedilmişse, Static Agent, içinde bulunduğu Jade container'ın adresini, alarma sebep olan sızmanın tipi ve zamanını içeren bir mesajı MainAgent'a gönderir ve bu sızmaya ait test zamanını başlatır.
- MainAgent, Static Agent'lardan gelen sızma şüphesi mesajlarını alır. Her ayrı tipteki sızma için, ayrı birer Saldırılan Konakçı Listesi (Victim Host List-VHL) oluşturur. Eğer bir saat içerisinde farklı bir konakçıdan aynı tip saldırı şüphesi mesajı gelirse, o mesaj bilgilerini ilgili VHL'e ekler; aksi takdirde o tipte yeni VHL yaratır. VHL eleman sayısı iki olduğunda, ilgili tipteki sızmaları araştıracak nitelikte bir MobileAgent yaratır, VHL'ı aktarır ve Main Container'da başlatılmasını sağlar.
- Mobile Agent, kendi VHL'sindeki ilk konakçıya gider ve buradaki Static Agent'a, gönderdiği sızma şüphesi mesajıyla ilgili detaylı bilgi isteyen bir mesaj gönderir.

- Static Agent, ilgili alarma sebep olan kaynağın IP adresini kendi "alert.ids" log dosyasından öğrenerek Mobile Agent'a cevap mesajı olarak gönderir. Log'daki ilgili alarm satırını, baştan itibaren arayarak değil de, Mobile Agent'tan aldığı satır numarasından yararlanarak kolayca bulur.
- Mobile Agent, birinci konakçıdan aldığı IP adresi ile birlikte listesindeki ikinci konakçıya gider. Sıradaki konakçıda yer alan Static Agent'a kendisindeki IP numarasını verir ve kendi log dosyasındaki kayıtlarla karşılaştırmasını ister. Bu bilgiyi alan Static Agent, kendi alarm dosyasındaki IP numaralarıyla Mobile Agent'tan aldığı IP numarasını karşılaştırır. Static Agent, ilgili alarm dosyasını baştan itibaren tarar.
- Eğer IP numaraları aynı ise, dağıtık bir saldırı tespit edilmiş olur. Bu durumda Mobile Agent ilgili dağıtık saldırıya ait bilgileri AlarmAgent'a bir mesaj halinde gönderir ve kendisini yok eder. Konakçıdan IP numaralarının eşleşmesi konusunda olumsuz cevap alınırsa, Mobile Agent sıradaki diğer konakçıda aynı işlemleri tekrarlar. Eğer aynı IP numaralı ikinci bir saldırı şüphesi bulunamazsa aynı işlemler bir saat boyunca tekrarlanır ve sonunda Mobile Agent kendisini yok eder.
- AlarmAgent gelen mesajdaki bilgileri kullanarak, Güvenlik Konsoluna uyarı mesajı yazdırır ve ilgili değerlendirmeleri yapmak amacıyla bir kayıt dosyası oluşturur.

3.2.2 Alarm dosyalarının Snort yardımıyla oluşturulması

WinIDS, Snort yazılımının Windows işletim sisteminde MySQL veritabanı ve Apache web sunucusu ile beraber kullanıldığı bir sızma belirleme sistemidir. Bu sistem içinde kullanılan programlar ve kısa açıklamaları şu şekildedir:

- Apache Web Sunucusu: BASE adlı web-tabanlı güvenlik konsoluna sunuculuk yapmaktadır.
- Snort: Ağdan bilgi toplamak ve bunları analiz etmek için kullanılan sızma belirleme sistemidir.
- WinPcap: Windows işletim sisteminde ağ kartından veri toplamaya ve karta veri göndermeye; bu verileri süzmeye ve bir tampon alanda saklamaya yarayan ağ kartı sürücüsüdür.

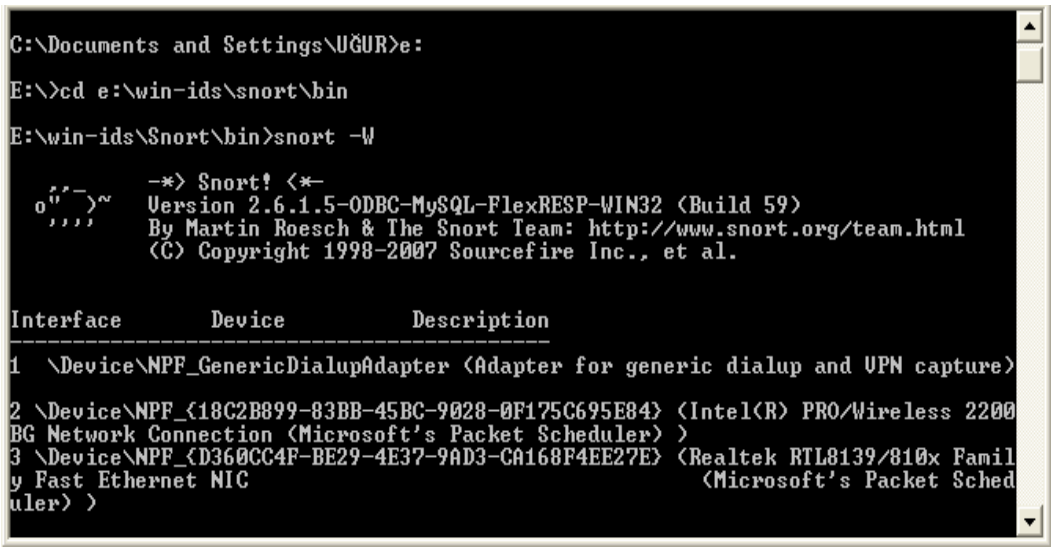
- MySQL Sunucu: Alıcılar tarafından tetiklenen tüm sızma alarmları MySQL veritabanında saklanmaktadır.
- PHP: BASE adlı dinamik web sayfasını geliştirmek için kullanılmıştır.
- BASE (Basic Analysis and Security Engine): Snort tarafından oluşturulan sızma alarmlarını görüntüleyen web-tabanlı uygulamadır. Alıcı bilgilerinin tamamı birleştirilir ve ilişkilendirilir [68].

Kurulum öncesinde, WinIDS bileşenlerinin kurulacağı sürücüde WinIDS klasörü oluşturuldu. Apache sunucusu ile aynı ortamda olmaması gereken IIS (Internet Information Services) sunucusunun kurulu olmadığı teyit edildi. 'C:\WINDOWS\system32\drivers\etc\host' dosyasına '127.0.0.1 winIDS' ifadesi eklenerek "winIDS"in de "localhost" gibi lokal olarak çağrılabilmesi sağlandı. WinPcap ve Snort kuruldu. 'snort.conf' adlı konfigürasyon dosyasında aşağıda sıralanan değişiklikler yapıdı:

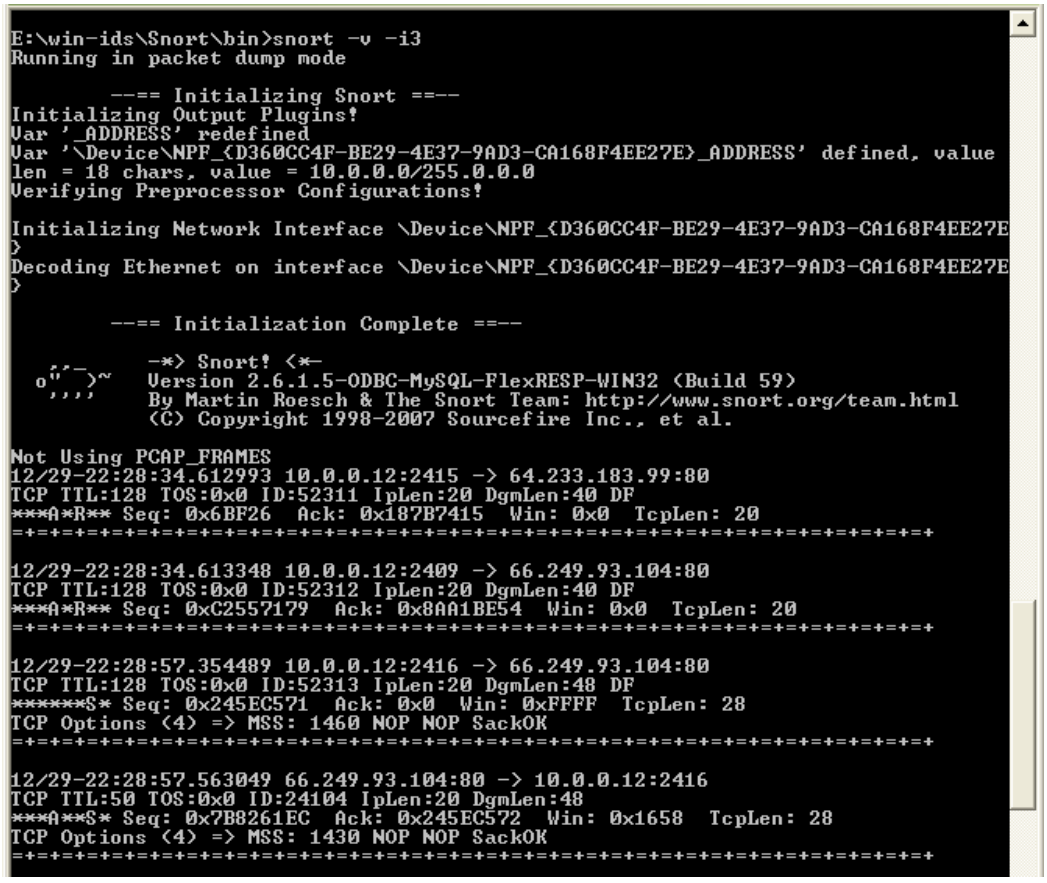
- Verisetinde yer alan konakçıların her birisinin sabit IP numaraları ilgili web sayfasında verilmiştir. Ağdaki tek bir noktada toplanılan veriseti trafiğini her bir konakçıda ayrı ayrı toplanmış gibi modellemek için, konakçı adedince Snort koşturuldu. Her defasında Snort'un saldırılardan koruyacağı bilgisayarın IP numarası "var HOME_NET any" ifadesindeki "any" yerine yazılarak Snort'un tüm ağda değil sadece ilgili konakçıda sızma belirleme yapması ve verisetinin ayrıştırılması sağlandı.
- "var EXTERNAL_NET any" ifadesinin yerine "var EXTERNAL_NET !$HOME_NET" yazıldı.
- "var RULE_PATH ../rules" ifadesinin yerine "var RULE_PATH e:\win-IDS\snort\rules" yazılarak Snort'a ait sızma belirleme kurallarının yeri gösterildi. Kural dosyalarındaki "ANY" ifadelerinin yerine "$HOME_NET" yazıldı.
- "Preprocessor sfportscan" bölümündeki "watch_ip{}" ifadesinde süslü parantezler içerisine sızma belirleme yapılacak olan konakçının IP numarası yazıldı.
- "output alert_fast: alert.IDS" ifadesi eklenerek sızma alarmlarının bir log dosyasında kaydedilmesi sağlandı.

- “output database: log, mysql, user=snort dbname=snort host=localhost sensor_name=WinIDS” değişikliği eklenerek alarmların veritabanına da kaydedilmesi sağlandı.

Şekil 2.2 ve Şekil 3.3’de Snort’un çalıştırılması görüntülenmiştir.



Şekil 3.2 : “snort –W” komutu.



Şekil 3.3 : “snort –v –i3” komutu.

Apache web sunucusu kuruldu. 'httpd.conf' konfigürasyon dosyasında aşağıdaki değişiklik yapıldı:

- “#LoadModule ssl_module modules/mod_ssl.so” ifadesinin altına “LoadModule php5_module e:\win-IDS\php\php5apache2_2.dll”, “AddType application/x-httpd-php .php”, “PHPIniDir e:\win-IDS\php” ifadeleri eklenerek Apache sunucusunun PHP’yi çalıştırması sağlandı.

“e:\win-IDS\php” içerisine PHP kuruldu. “c:\windows\system32” içerisine “e:\win-IDS\php\” içerisindeki “libmysql.dll” dosyası kopyalandı. “e:\win-IDS\php\php.ini” dosyasında aşağıda sıralanan değişiklikler yapıldı:

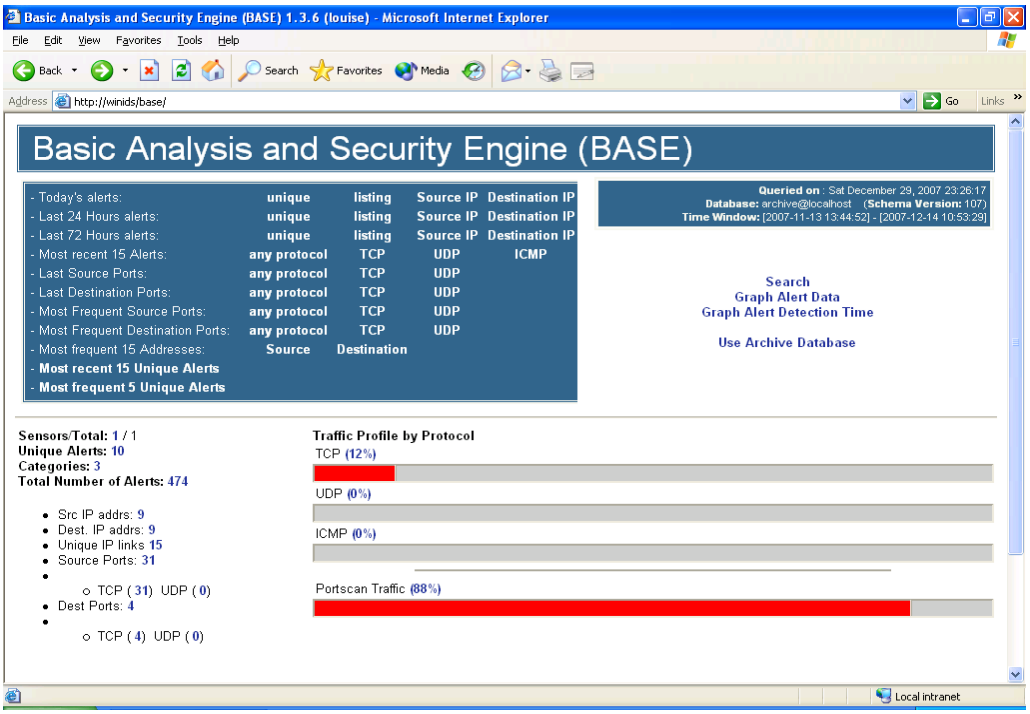
- “extension_dir = "./"” ifadesi yerine “extension_dir = "e:\win-IDS\php\ext"” ifadesi yazıldı.
- “;extension=php_gd2.dll” ve “;extension=php_mysql.dll” ifadelerinin başındaki “;” işareti kaldırıldı.
- “;session.save_path = "/tmp"” ifadesi yerine “session.save_path = "c:\windows\temp" ” ifadesi yazıldı.

MySQL kuruldu. Snort veritabanları ve tabloları yaratıldı. Veritabanı erişiminin güvenliği sağlandı.

BASE Güvenlik Konsolu kuruldu. “e:\win-IDS\snort\doc\signatures” klasörü “e:\win-IDS\apache\htdocs\base\” içerisine kopyalanarak saldırılara ait imzalar tanıtıldı. “e:\win-IDS\apache\htdocs\base\base_conf.php” konfigürasyon dosyasına aşağıda sıralanan ifadeler eklendi:

- “$BASE_urlpath = 'http://winIDS/base';”
- “$DBlib_path = 'e:\win-IDS\adodb';”
- “$DBtype = 'mysql';”
- “$alert_dbname = 'snort';”
- “$alert_host = 'localhost';”
- “alert_user = 'base';”
- “$portscan_file = 'e:\win-IDS\snort\log\portscan.log';”

Şekil 3.4’de BASE’in çalıştırılması görüntülenmiştir.



Şekil 3.4 : BASE Güvenlik Konsolu.

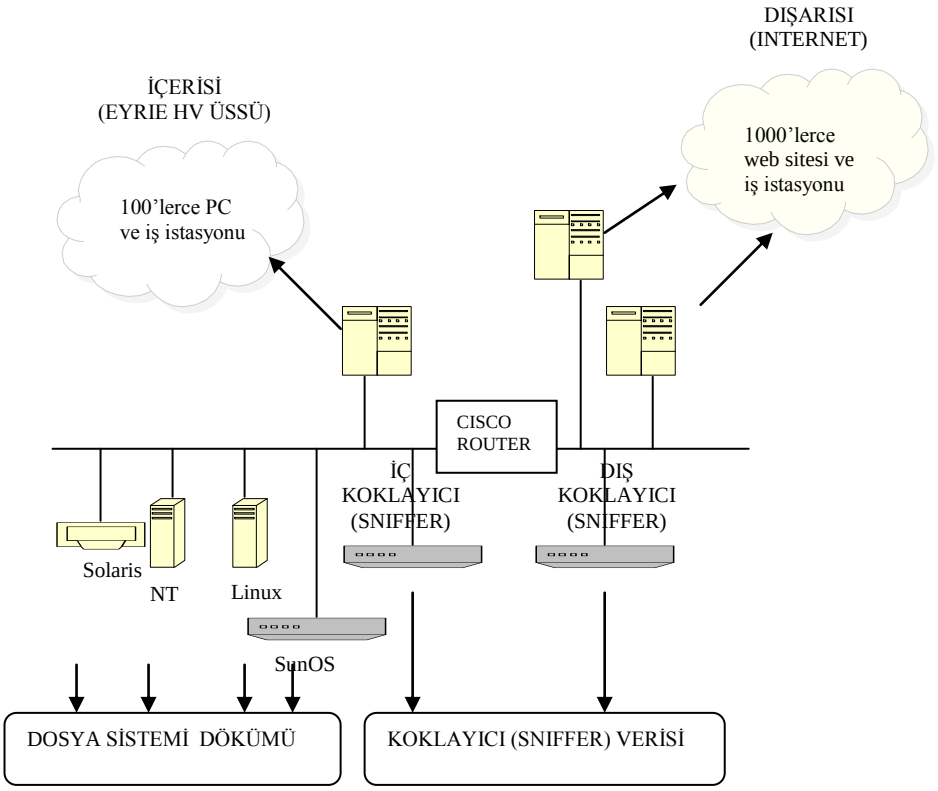
3.2.3 Kullanılan veriseti

Sızma belirleme değerlendirmeleri verilerinin açıklanması kişiye özellik kaygılarından dolayı kolay olmamaktadır. Bu sorunun üstesinden gelebilmek için, MIT Lincoln Laboratuvarı DARPA sponsorluğu altında, bir denektaşı olan Sızma Belirleme Değerlendirme Verisetini (IDEVAL) oluşturdu [69].

1998, 1999 ve 2000'de, bir Hava Kuvvetleri üssünü simule edecek şekilde bir ağ oluşturdular. Arkaplan aktiviteleri üretildikten ve saldırılar iyi tanımlanmış noktalara yerleştirildikten sonra tcpdump, Sun BSM, proses ve dosya sistemi bilgilerini topladılar. Blok diyagramı Şekil 3.5'de gösterilen test verisinde 58 farklı saldırının 200'den fazla örneği yer almaktadır [70].

DARPA için oluşturulan ilk saldırı senaryo örneği, bir DDoS saldırısı içeren 2000 yılındaki LLS_DDOS 1.0 verisetidir. Bu saldırının temeli, bir saldırıcının internetteki çok sayıdaki konakçıya girmek, bir DDoS saldırısı çalıştırmak için gerekli bileşenleri yüklemek ve bir ABD devlet sitesine DDoS saldırısı başlatmak için betik bir saldırı kullanarak kendi becerisini gösterme arayışıdır. Saldırıcı, simule ağdaki üç Solaris konakçıya yönetici erişimi sağlamak için Solaris sadmind açığını, saldırıyı başlatmak için de Mstream DDoS aracını kullanmaktadır. Kötüye kullanılan üç ara konakçıya da birer Mstream "server" kurulurken, bunlardan bir tanesine "server"ları kontrol

eden bir Mstream "master" kurulmuştur. DDoS sadırısı bu “server”lar tarafından eşzamanlı olarak başlatılmıştır [71].



Şekil 3.5 : 1999 test-yatağının blok diyagramı.

Saldırı senaryosunun beş aşaması vardır:

1. Ağın IPsweep ile taranması ,
2. Solaris konakçılarda çalışan sadmind aracını aramak için aktif konakçıların incelenmesi,
3. Sadmind açığını kullanarak konakçılara girilmesi,
4. Trojan mstream DDoS yazılımının üç konakçıya da yüklenmesi,
5. DDoS saldırısının başlatılması.

Bu veri dosyaları, 7 Mart 2000 günü 9:25 AM ile 12:35 PM arasında yaklaşık üç saat boyunca oluşturulmuştur. Saldırı senaryosunun beş aşaması detaylı olarak aşağıda anlatılmıştır:

1’inci Aşama: 202.77.162.213 IP numaralı saldırıcı, aktif konakçıları belirlemek için simule Hava Kuvvetleri Üssündeki, toplam 20 konakçı içeren bazı C sınıfı alt ağlara

(172.16.115.0/24 gibi) ICMP echo-istekleri gönderen ve ICMP echo-cevaplarını dinleyen IPsweep saldırısı düzenler.

2'nci Aşama: Aktif konakçılar, kendilerinde "sadmind" uzaktan yönetim aracı kurulu olup olmadığının belirlenmesi amacıyla "sadmind"dan faydalanma (exploit) programının "ping" opsiyonu kullanılarak yoklanırlar (probe).

3'üncü Aşama: "sadmind" aracı kurulu olan aktif konakçılara, yönetici erişimi elde etmek ve sisteme izinsiz girmek amacıyla saldırı düzenlenir. Bu saldırılardan sonra, saldırıcı, izinsiz girişlerin başarılı olup olmadığını test etmek için telnet aracılığıyla bir oturum açma girişiminde bulunur.

4'üncü Aşama: Bu aşamaya kadar mill (172.16.115.20), pascal (172.16.112.50) ve locke (172.16.112.10) adlı konakçıların yönetici erişimleri elde edilmiştir. Saldırıcı bu konakçıların tamamını "daemon" ve bir tanesini "master" olarak belirledikten sonra, konakçılar kendi aralarında gerekli mesajlaşmaları yaparlar.

5'inci Aşama: Bu son aşamada, saldırıcının "mstream 131.84.1.31 5" komutuyla beraber "daemon"lar tarafından eş zamanlı olarak 131.84.1.31 IP adresli hedef konakçıya 5 saniye süreli DDoS saldırısı başlatılır. Gönderilen mstream DDoS paketlerinin hiçbirinin kaynak IP adresleri doğru değildir [71].

Deneylere başlamadan önce iç sensör görevi yapan konakçının topladığı çevrim-dışı ağ trafiği verisi tcpdump dosyası olarak indirildi. İç sensör, bütün ağ trafiğini diğer konakçılara ve dışarıya gönderilmeden önce tek bir noktada toplayabilmek için ateş duvarının hemen arkasına yerleştirilmiştir. Tek dosya halindeki tcpdump verisi, Snort kullanılarak, her biri bir konakçıya ait olan yirmi farklı "alarm.ids" dosyası halinde çözümlenmiştir. Böylece, veri setindeki olası sızma olaylarına ait Snort alarmları elde edilmiştir. DDoS saldırı aşamalarındaki sızma tipleri Snort alarmlarında,

- 1'inci Aşama: ICMP PING
- 2'inci Aşama: RPC portmap sadmind request UDP, RPC sadmind UDP Ping
- 3'üncü Aşama: RPC sadmind query with root credentials attempt UDP, RPC sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attack
- 4'üncü Aşama: RSERVICES rsh root

- 5’inci Aşama: telnet to master (DDoS saldırısı başlatmak için)

olarak görünmektedirler. Çizelge 3.1’de saldırı aşamalarının gerçekleşme zamanları (Türkiye saat diliminde) ve kaç adet konakçıya yapıldıkları detaylı olarak yer almaktadır.

Çizelge 3.1 : Saldırıların gerçekleşme zamanları.

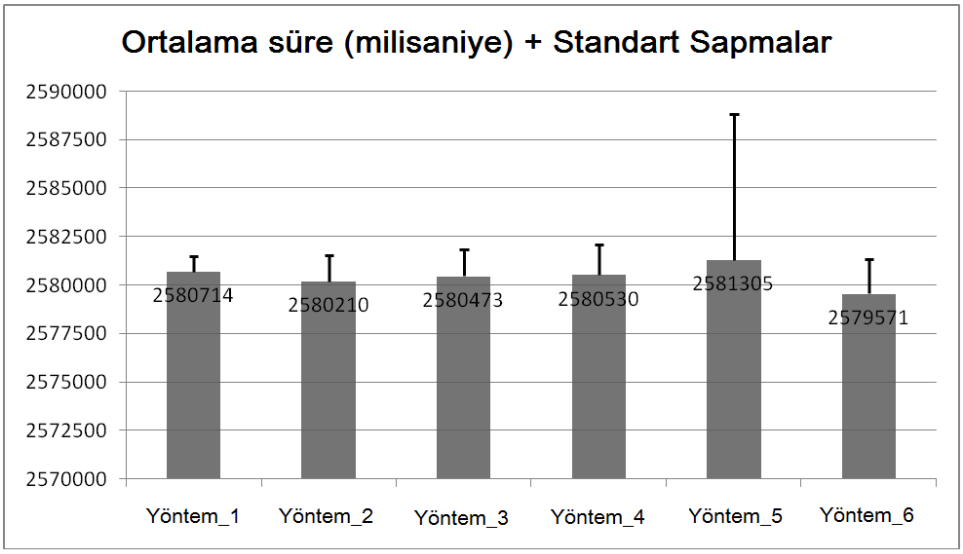
Saldırı Aşamaları	**Saldırı adları**	**Saldırı yapılan konakçı sayısı**	**Saldırının ilk konakçıya yapıldığı zaman**	**Saldırının ikinci konakçıya yapıldığı zaman**	**Saldırının son konakçıya yapıldığı zaman**
1 (ICMP)	ICMP PING	20	16:51:36.14	16:51:36.52	16:52:00.82
2a (UDP)	RPC portmap sadmind request UDP	11	17:08:07.35	17:08:07.50	17:34:36.43
2b (UDP)	RPC sadmind UDP Ping	3	17:08:07.36	17:15:10.03	17:15:10.10
3a (UDP)	RPC sadmind query with root credentials attempt	3	17:33:10.62	17:34:36.45	17:34:49.00
3b (TCP)	RPC sadmind UDP NETMGT_PROC SERVICE CLIENT_DOMAIN overflow attack	3	17:33:10.62	17:34:36.45	17:34:49.00
MİNİMUM SALDIRI TESPİT SÜRESİ = 42 DAKİKA 59.93 SANİYE = 2579930 MSN					
4	RSERVICES rsh root		17:50:02		17:50:38

Dağıtık bir saldırının varlığını tespit edebilmek için, bu senaryonun ilk üç aşamasının belirlenmesi ve son iki aşaması gerçekleşmeden önce güvenlik yöneticisinin haberdar edilmesi hedeflenmiştir. Bu doğrultuda, ilk üç aşamanın MA’lar tarafından belirlenebilmesi ve DDoS saldırısının varlığına karar verilebilmesi için birinci aşamanın ilk konakçıya yapılmasından itibaren en az 42 dakika 59.93 saniyenin geçmesi gerektiği ortaya çıkmıştır. Bu durum, aşağıdaki formülde (3.1) de ifade edilmektedir:

DDoS kararı verebilmek için gereken min. Süre = 2’inci konakçının 3’üncü aşamaya maruz kaldığı zaman - 1’inci konakçının 1’inci aşamaya maruz kaldığı zaman (42 dakika 59.93 saniye = 18:34:36.45 - 17:51:36.14) **(3.1)**

3.2.4 Sonuçlar

Her bir yöntemin 20'şer kere koşturulması sonucunda Şekil 3.6'daki sonuçlar elde edilmiştir. Yukarıda belirtildiği gibi, minimum saldırı tespit süresinin 42 dakika 59.93 sn (2579930 msn) olduğu değerlendirildiğinde, bütün yöntemlerin ortalama saldırı belirleme sürelerinin, minimum süre (+1375, -359 msn) içerisinde olduğu görülmektedir. Ortalama süreler standart sapmalarıyla beraber değerlendirildiğinde, yöntemlerin saldırı belirleme süresi açısından birbirlerinden çok farklı olmadıkları ve minimum saldırı tespit süresine çok yakın değerlere sahip oldukları anlaşılmaktadır.



Şekil 3.6 : Simule-gerçek zamanlı testlerin belirleme sürelerine ait sonuçlar.

Dikkat edileceği gibi, saldırı aşamalarının tespit edilme süreleri saldırının ilk iki konakçıda gerçekleşme zamanları arasındaki süreden daha küçük değerler alabilmektedir. Çünkü bu ölçümler, saldırıların ilk iki konakçıda gerçekleşme zamanları arasındaki farkı değil, "dağıtık saldırının yapıldığı uyarısının AlarmAgent'a ulaştığı zaman" ile "saldırının ilk konakçıya yapılması sonrasındaki işlemlerin başlatıldığı zaman" arasındaki farkı ortaya koymaktadır. Bu işlemler, ilk üç yöntemde, ilk mesaj MainAgent'a ulaştığı zaman, sonraki yöntemlerde ise saldırının ilgili konakçıdaki SA tarafından belirlendiği zaman başlatılmaktadır.

Çizelge 3.2'de yöntemlerin ağ yükü ölçümleri ve bu ölçümlere göre olan sıralamaları verilmiştir. Ağ yükü, konakçılar arasında gönderilen mesajlar ve ağda dolaşan Mobile Agent'lar tarafından oluşturulmaktadır. Çizelgede, bu mesaj ve Mobile Agent'ları temsil etmesi için kısaltmalar kullanılmıştır. Örneğin, Yöntem_3'te Static Agent'lar tarafından MainAgent'a gönderilen mesajlar Mes3_SAtoMainA ifadesiyle,

Yöntem_5'te Mobile Agent'lar tarafından tüm konakçılara gönderilen mesajlar Mes5_MAtoALL ifadesiyle ve Yöntem_4'deki Mobile Agent'lar MA_4 ifadesiyle temsil edilmektedir.

Çizelge 3.2 : Ağ yükü ölçümleri.

	Yöntem_1	**Yöntem_2**	**Yöntem_3**	**Yöntem_4 (MainAgent 10% çökmüştür)**	**Yöntem_5**	**Yöntem_6 (MainAgent 10% çökmüştür)**
Mes1_SAtoMainA (3080 bit)	236					
Mes2_SAtoMainA (3250 bit)		236				
Mes3_SAtoMainA (3250 bit)			236			
Mes4_SAtoMainA (3080 bit)				56		
Mes4_MainAtoSA (850 bit)				50		
Mes5_MAtoALL (490 bit)					210	
Mes6_SAtoMainA (3250 bit)						215
Mes6_MainAtoSA (1150 bit)						212
Mes6_MAtoALL (490 bit)						23
Mes6_SAtoALL (550 bit)						2
MA_2 (1050000 bit)		10				
MA_3 (1500000 bit)			16			
MA_4 (750000 bit)				18		
MA_5 (750000 bit)					52	
MA_6 (1020000 bit)						14
TOPLAM(KB)	88,73	1375,37	3023,32	1674,19	4773,3	1888,12
SIRALAMA DERECELERİ	Sınıflandırılmadı	1	4	2	5	3

Yöntem_2, her biri 3250 bit büyüklüğünde olan 236 adet Mes2_SAtoMainA mesajı ve her biri 1050000 bit büyüklüğünde olan 10 adet MA_2 gezgin etmenin toplamı olarak 1375,37 KB ağ yüküne sebep olmuştur. Mesajların ve Mobile Agent'ların parantez içerisinde verilen büyüklük değerleri taşıdıkları yük ile beraber değişmektedir. Bu çalışmadaki testlerde mesajların taşıdıkları yük miktarları küçüktür, çünkü sızma-şüpheleri için yeterli ama az miktarda bilgi taşımaktadırlar.

Testlerin %10'unda yöntem_4 ve yöntem_6'nın MainAgent'ı çalışamaz duruma getirilmiştir ve sistem MainAgent olmadan çalışmıştır. Yöntemlere ait olan mesajların ve Mobile Agent'ların sayısı ilgili yöntemin sütununda verilirken, toplam

büyüklük en sonda belirtilmiştir. Yöntemlerin sıralamadaki yerleri, ağ yükü büyüklükleri ile doğru orantılıdır, çünkü en iyi yöntemin en az ağ yükü değeri olmalıdır. Yöntem_1, bu çalışmanın ilgi alanı dışında kalacak şekilde merkezi üniteye tam bağımlı olduğu için, bu ve sonraki sıralamalarda sınıflandırmaya katılmamıştır.

Önceden de belirtildiği gibi, MainAgent tarafından bir Mobile Agent yaratmak için yöntem_2'de iki adet sızma-şüphesi mesajı gerekirken, yöntem_3'te bir tanesi yeterli olmaktadır. Bu iki yöntemin Mobile Agent'ları MainAgent tarafından yaratıldıktan sonra merkezden bağımsız olarak çalışabilmektedirler. Son üç yöntemdeki Mobile Agent'lar herhangi bir zamanda MainAgent'ın çalışamaz olması durumunda dahi bağımsız olarak çalışabilmektedirler. Bu "merkezi üniteden bağımsızlık" durumları, Çizelge 3.3'de görülebilmektedir. "1" değeri, ilgili yöntemdeki sistemin belirtilen koşullarda MainAgent'ın göçmesi durumunda bile çalışabildiği anlamına gelmektedir. Yöntemlerin sıralama dereceleri toplam değerlerle ters orantılıdır, çünkü en iyi yöntemin bağımsız olduğu durumlar en çok olmalıdır. Bu toplam değer, "i" değişkeninin yöntem numarasını temsil ettiği "$toplam_i$ = $değer_{i1}$+ $değer_{i2}$ + $değer_{i3}$" formulüne göre hesaplanmaktadır.

Çizelge 3.3 : Merkezi üniteden bağımsızlık.

	MainAgent herhangi bir zamanda kullanılamaz olur	**MainAgent bir sızma-şüphesi mesajı kabul ettikten ve Mobile Agent yarattıktan sonra kullanılamaz olur**	**MainAgent iki sızma-şüphesi mesajı kabul ettikten ve Mobile Agent yarattıktan sonra kullanılamaz olur**	**Toplam**	**Sıralama Dereceleri**
Yöntem_1	0	0	0	0	-
Yöntem_2	0	0	1	1	3
Yöntem_3	0	1	1	2	2
Yöntem_4	1	1	1	3	1
Yöntem_5	1	1	1	3	1
Yöntem_6	1	1	1	3	1

Simule-gerçek zamanlı test sonuçlarında Yöntem_6'nın Yöntem_5'e göre daha fazla ağ yükü dezavantajı ortaya çıkmıştır; çünkü MainAgent mod-1'deyken, mod-2'ye geçişi kontrol edebilmek amacıyla, kendisine gelen her mesaja karşılık cevap mesajı göndermektedir.

Yöntemlerin iki kritere göre olan genel sıralaması Çizelge 3.4'de görülmektedir. Tüm yöntemlerde benzer değerler olduğu için "sızma belirleme süresi" üçüncü bir kriter olarak kullanılmamıştır. Ortalama değerleri hesaplandıktan sonra sıralama dereceleri normalize edilmiştir. Bu derecelere göre, Yöntem_4 en iyi yöntem olarak ortaya çıkmaktadır.

Çizelge 3.4 : Simule-gerçek zamanlı testlerdeki yöntemlerin sıralaması.

	"Ağ yükü" dereceleri	**"Merkezi üniteden bağımsızlık" dereceleri**	**Derecelerin ortalaması**	**Normalize edilmiş dereceler**
Yöntem_2	1	3	2	2
Yöntem_3	4	2	3	3
Yöntem_4	2	1	1.5	**1**
Yöntem_5	5	1	3	3
Yöntem_6	3	1	2	2

4. YAPAY BAĞIŞIKLIK SİSTEMİNDEN ESİNLENEN ÇOK-AMAÇLI EVRİMSEL ALGORİTMA KULLANAN ANOMALİ-TEMELLİ SIZMA BELİRLEME

Bir önceki bölümde geliştirilen yöntemlerde, imza-temelli bir ağ sızma belirleme sistemi olan Snort tarafından yaratılan alarm verileri kullanılmıştır. İmza-temelli SBS'lerinin yeni saldırıları belirleme konusundaki başarısızlığının üstesinden gelebilmek için, tasarladığımız sisteme uygun ve amaca yönelik verisetlerine yüksek oranlarda başarılı cevaplar verecek bir anomali-temelli SBS geliştirme ihtiyacı ortaya çıkmıştır. SBS mimarisi ve Biyolojik Bağışıklık Sistemi arasındaki benzerlikten dolayı, bir anomali-temelli SBS yöntemi olarak Yapay Bağışıklık Sistemini (YBS) kullanılmıştır.

4.1 Sistem Tasarımı

DDoS saldırılarını belirlerken daha iyi doğru ve yanlış pozitif oranları elde edebilmek için bir Çok-amaçlı Evrimsel Algoritmadan esinlenen Yapay Bağışıklık Sistemi olan jREMISA (Java REtrovirus-inspired Multiobjective Immune System Algorithm) çalışması iyileştirildi. Genel konsept korunarak, r-sürekli değerlendirmeler yöntemi [38] eklendi, Negatif Seleksiyon ve Klon Seleksiyon yapısı değiştirildi ve hedefler yeniden tanımlandı. Geliştirilmiş-jREMISA, MIT DARPA LLS_DDOS 1.0 ve DARPA 1999 verisetlerinde ayrı ayrı test edildi ve sonuçlar literatürdeki diğer çalışmalarla karşılaştırıldı.

4.1.1 jREMISA çalışması

Yapay bağışıklık sisteminin [42] çok-amaçlı evrimsel algoritma [64] ile beraber kullanıldığı bu çalışmada [5] en iyi sınıflandırma uygunluk derecesi ve çok-amaçlı hiper-hacim büyüklüğüne sahip belirleyiciler hedeflenmektedir. Sızma belirleme sistemi yapısının insan vücudundaki antijenleri belirleyen ve yok eden paralel ve dağıtık bir adaptif sistem olan biyolojik bağışıklık sistemine benzemesinden dolayı, sızma belirleme yöntemi olarak yapay bağışıklık sistemi seçilmiştir. AIS-temelli

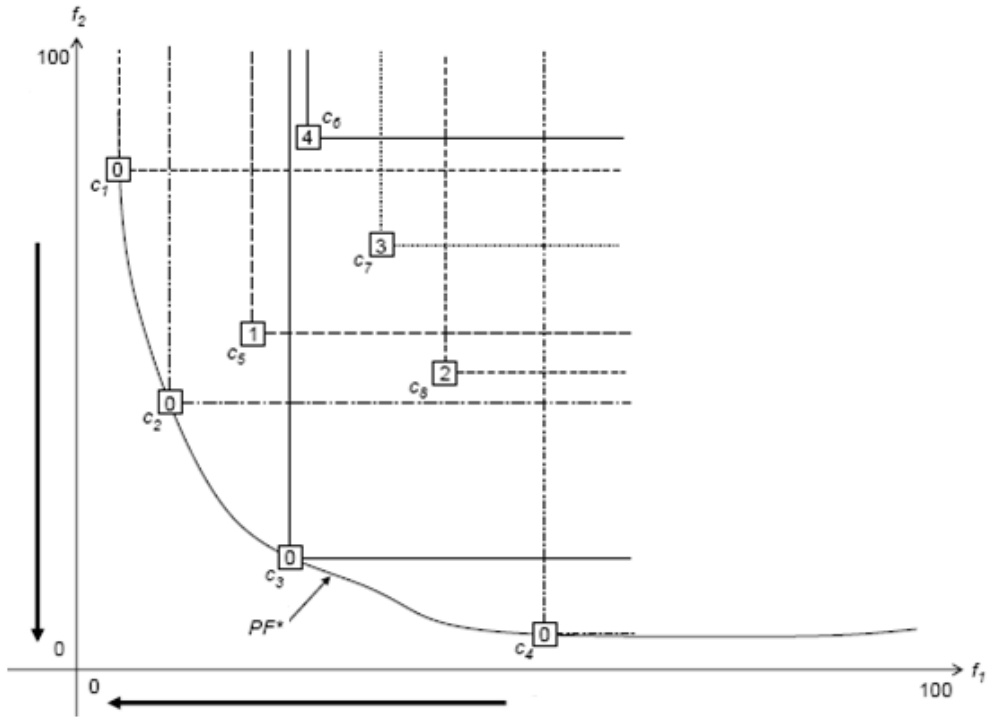
SBS'inde ağ trafiği, antijen belirleyicilerin (antikor) bir öğrenme verisetine dayanarak geliştirilmesi sayesinde, kendinden-olan veya kendinden-olmayan olarak sınıflandırılmaktadır.

Bir hata değeri kapsamında, ikilik değerlerden oluşan string ifadelerin karşılaştırılması sonucunda ideal sonuca polinom zamanda ulaşılamadığı için, YBS'ne, içerdiği stokastik iyileştirme yöntemi sayesinde polinom zamanda kabul edilebilir sonuçlar veren çok-amaçlı evrimsel algoritma eklenmiştir. MOEA, verilerin birden fazla amaca yönelik değerlendirilmesi sonucunda karar-vericiye tek bir çözüm yerine ödünleşim (trade-off) çözümler kümesi sunduğu için tercih edilmiştir. Bu amaçlar:

1. *Mümkün olan en yüksek doğru sınıflandırma oranını elde etmek:* Doğru pozitif değerlendirmelerde antikorların (Ab) antijenlerden (Ag) farklı olan bitlerin sayısını ve doğru negatif değerlendirmelerde aynı olan bitlerin sayısını toplayarak elde edilen sınıflandırma hata oranı minimize edilmek istenmektedir. Çünkü, bu hedefin toplam skoru azaldığında belirleyicinin verimliliği artmaktadır.
2. *Belirleyicilerin hiper-hacmini benzerlik eşiğine (affinity treshold) göre maksimize etmek:* Hiper-hacim, Pareto-front noktalarının bir referans noktasıyla beraber hedef uzayında kapsadıkları diktörgen alan olup çözümlerin kalitesini gösteren bir ölçümdür. Belirleyicilerin önceden belirlenen negatif-seleksiyon benzerlik değerinden çok fazla sapmamaları neticesinde düşük sapma değerleri istenilmektedir. Belirleyicilerin kapsamı, ne normal trafiğe anomali diyecek kadar yüksek; ne de anomali tarafiğe normal diyecek kadar düşük olmamalıdır. Bu nedenle belirleyicinin kapsamını belirleyen benzerlik-eşiğinden-olan-sapmaların mümkün olduğunca sıfıra yakın olması istenilmektedir.

Pareto Optimumluğu (Pareto Optimality-P*) kullanılarak üstün (nondominated) çözümler elde edilmek istenmektedir. Bir çözümün Pareto-optimum olması için (global-minimum aranıyorsa) her iki hedef değerleri de diğer çözümlerin değerlerinden ya daha düşük ya da eşit olmalı ve hedef değerlerinden en az birisi diğer çözümlere ait aynı hedef değerlerinden daha düşük olmalıdır. Pareto-optimum çözümler kümesine Pareto-front (PF*) denilmektedir.

Şekil 4.1'deki çözüm noktalarının üzerinde diğer çözüm noktalarından toplam kaç tanesinin kendisinden-üstün olduğunu gösteren rakam vardır. Pareto Front çizgisi üzerindeki noktaların değeri her zaman sıfırdır, çünkü hepsi üstün çözümlerdir. Diğer noktaların değeri en küçük bir olacaktır. Bir çözüm noktasının üstün olabilmesi için bütün amaçlar için diğer çözümlerden daha optimum değere sahip olması gerekmektedir.



Şekil 4.1 : İki amaçlı minimizasyon problemine ait Pareto front.

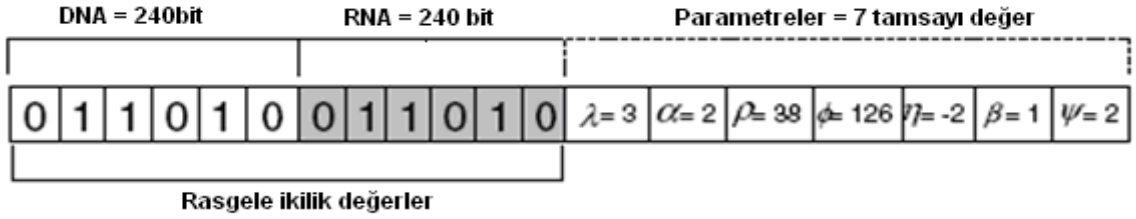
Sonuç olarak, çözümlerin kalitesini gösteren, hedef uzaydaki Pareto-Front noktaları ve bir referans noktası tarafından örtülen dikdörtgen alan olan, belirleyici hiper-hacminin maksimize edilmesi amaçlanmaktadır.

4.1.1.1 Antijen ve Antikorların temsil edilmesi

Antijen (Ag) ve Antikor (Ab) kromozomları ikilik dizilerdir. Antijenler, en yaygın üç IP protokol olan TCP, UDP ve ICMP trafikleri için farklı temsil edilmektedir. IP, TCP, UDP ve ICMP başlıklarının bütün olası alanlarını temsil edecek şekilde (IP=122, TCP= 118, UDP=48, ICMP=16), TCP Ag 240 bit, UDP Ag 170 bit, ICMP Ag 138 bit kullanılarak kodlanmıştır. Ag paket başlıklarının ondalık değerleri denk ikilik değerlere dönüştürülmüştür.

Ab kromozomları Şekil 4.2'de gösterildiği gibi DNA (iklik), RNA (ikilik) ve yedi adet durum özelliği (tamsayı) olmak üzere üç parçadan oluşmaktadır. Ag kromozomlarına karşı değerlendirilmesi yapılan tek parça olan DNA bitleri Negatif Seleksiyon sırasında yaratılırlar. RNA, DNA'nın bir kopyasıdır ve lokal optimumdan kaçmak için kullanılır. Eğer mutasyon sonrası DNA'nın uygunluk değeri mutasyon öncesi DNA'dan daha iyiyse yeni DNA RNA'ya kopyalanır; değilse eski gen değerlerine dönebilmek için RNA DNA'ya kopyalanır. Kromozomların sonunda yedi karakter vardır: λ=isim, α=yanlış belirleme sayısı, ρ= (yanlış pozitif + doğru negatif) uygunluk skoru, φ= (yanlış pozitif + yanlış negatif) uygunluk skoru, η=benzerlik eşiğinden olan sapma, β=ağda yayımlanıp yayımlanmadığı (evet/hayır), ψ=bu Ab'ye baskın gelen Ab'lerin sayısı. DNA yapısı ikilik değerli bitlerden oluştuğu için benzerlik ölçümü olarak, formulü aşağıda verilen (4.1) Hamming uzaklığı tercih edilmiştir.

$$H = \sum_{i=1}^{L} \delta_i \ , \ \delta_i = \begin{cases} 1 \ , \ Ab_i \neq Ag_i \\ 0 \ , aksi\ takdirde \end{cases} \qquad \textbf{(4.1)}$$



Şekil 4.2 : Ab Kromozomu [5].

4.1.1.2 Bağışıklık algoritması

jREMISA adlı çalışmanın algoritması Şekil 4.3'de verilmiştir. Ab'lerin hedef uzayda hareketi için mutasyonun yeterli olduğu ve iyi çözümleri bozabileceği düşüncesiyle çapraz-değişim (cross-over) uygulanmamıştır. Algoritmanın 3-7 satırlarında, Ab'lerin rastgele olarak yaratıldığı ve önceden belirlenen bir benzerlik eşiğine gore sadece kendinden-olan bir veriseti içerisindeki bütün Ag'lere karşı değerlendirildikleri Negatif Seleksiyon aşaması gerçekleştirilmektedir. Kendinden-olan bir Ag'ye benzerlik gösteren Ab'lerin yerine yenisi konulmadan elendiği aşamaya ait arayüz Şekil 4.4'de gösterilmektedir. Benzerlik eşiğinin %40 olması

durumunda, bir Ab kromozomunun kendinden-olan bir Ag'ye benzemesi nedeniyle elenmesi için Ab ile Ag arasındaki farklılığın en fazla %40 olması, yani benzerliğin en az %60 olması gerekmektedir. Benzerlik eşik değeri büyüdükçe bir Ab kromozomunun bir Ag kromozomuna benzeme ihtimali ve böylece elenme ihtimali artacaktır.

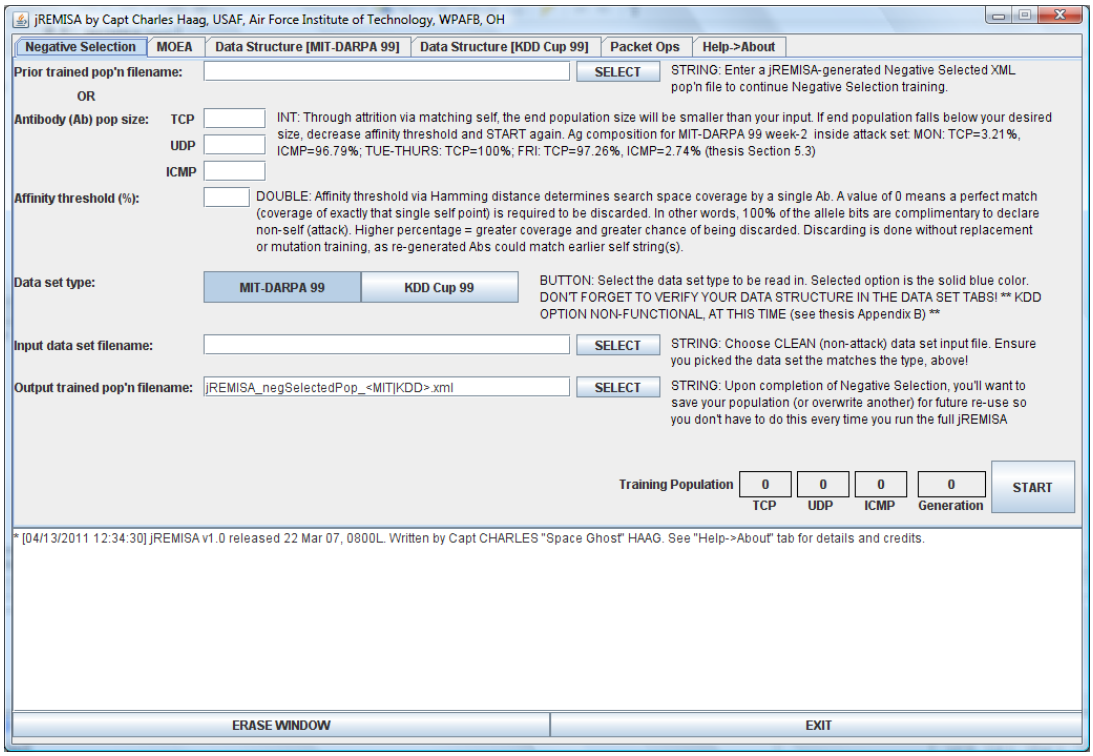
```
Procedure JREMISA
Begin
 Repeat
  Birincil TCP, UDP ve ICMP Popülasyon (Pop_p) yaratılması
  İkincil Popülasyon'un (Pop_s) boş olarak yaratılması
  Negative_selection(Pop_p,data_set_clean,threshold)
 until (end of data_set_clean)
 Repeat
  FitnessFunction (ag,threshold)
  MutationCauchy(Pop_p)
  P_optimality()
  ClonalSelection(0.05)
  MutationUniform(Pop_s)
  Pop_p ← Pop_s //En iyi Pop_s'i bir sonraki neslin Pop_p'una kopyala
    if (ağ iletişimi)
      broadcast(Pop_s)//İyi Ab'leri diğer AIS'lere öner
      processReceived()
    Endif
 until (end of data_set_attack)
End
```

Şekil 4.3 : jREMISA algoritması [5].

Birincil popülasyon, IP protokollerine gore üç gruba ayrılır; böylece TCP olmayan bir Ab TCP Ab'lerle karşılaştırılmamış olur. Değerlendirme penceresindeki her bir Ag, yeni bir nesli temsil eder ve aşağıda anlatılan işlemler populasyondaki Ab'lerin hepsine uygulanır:

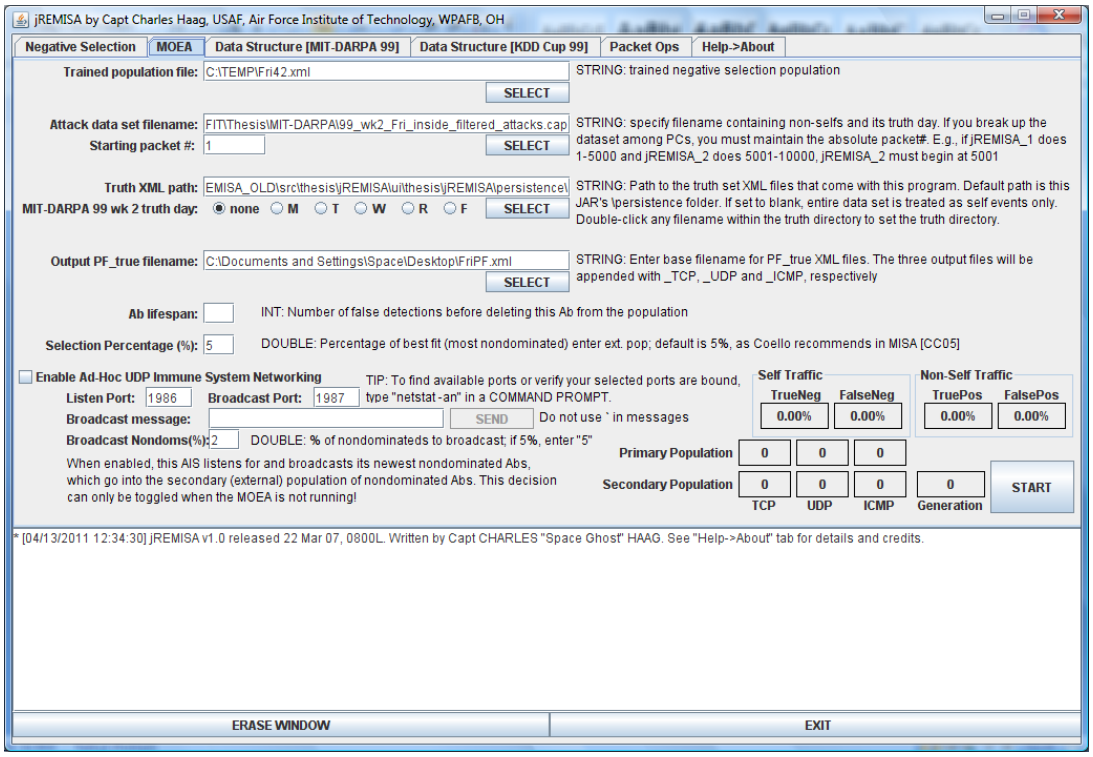
- Uygunluk Fonksiyonu: Ab ve Ag DNA genlerinin benzerliği olarak tanımlanan Hamming mesafesi (H) hesaplanır. Şekil 4.5'deki ÇAEA arayüzüne girilen benzerlik eşiği ve doğruluk-kümesi değerleri beraber değerlendirildiğinde aşağıdaki durumlardan birisi oluşacaktır:
    - Doğru negatif (Ag="kendinden-olan"dır, Ab de "kendinden-olan" olarak değerlendirir): obj1+= H, DNA'yı RNA'ya kopyala, obj2 += %1;

- Doğru pozitif (Ag="kendinden-olmayan"dır, Ab de "kendinden-olmayan" olarak değerlendirir): obj1+= (Ag uzunluğu – H), DNA'yı RNA'ya kopyala, obj2+= %1;
- Yanlış pozitif (Ag="kendinden-olan"dır, fakat Ab "kendinden-olmayan" olarak değerlendirir): yanlışBelirlemeler++, RNA'yı DNA'ya kopyala, obj2-= %1;
- Yanlış negatif (Ag="kendinden-olmayan"dır, fakat Ab "kendinden-olan" olarak değerlendirir): yanlışBelirlemeler++, RNA'yı DNA'ya kopyala, obj2-= %1.



Şekil 4.4 : jREMISA Negatif Seleksiyon arayüzü.

Ilk hedef doğru pozitif/negatif durumlarında, ikinci hedef ise yanlış pozitif/negatif durumlarında cazalandırılır. İdeal doğru negatif durumunda H değeri sıfır olmalıdır; aksi takdirde obj1 değerine aynı olan bitlerin sayısı eklenir. İdeal doğru pozitif durumda; H değeri Ag'nin uzunluğuna eşit olmalıdır; aksi takdirde obj1 değerine farklı olan bitlerin sayısı eklenir.



Şekil 4.5 : jREMISA ÇAEA arayüzü.

- Cezalandırılan Ab bitlerine Cauchy Mutasyon uygulanmaktadır. Cauchy dağılım, Ab'leri arama uzayında Gauss dağılımından daha geniş alanlara dağıtır.
- P*-Test: Her Ab, diğer Ab'lerin kaç tanesinin kendisinden üstün olduğunu belirlemek için Pareto-optimum testten geçer. Bu üstünlük rakamı kullanılarak bütün Ab'ler Quicksort algoritması ile sıraya dizilirler.
- Klon Seleksiyon: Üzerinde-başkalarının-üstünlük-kurmadığı (non-dominated) birincil popülasyon Ab'lerin %5'i elit seleksiyon ile seçilirler ve ikincil populasyona kopyalanırlar. Kopyalanan Ab'ler, geniş bir popülasyon oluşturmak için altı defa klonlanırlar. Kopyalanan ve klonlanan Ab'ler n-rastgele bit pozisyonu için mutasyona uğrarlar (n=hedef sayısı(2) + Pareto-baskınlık değeri). İkincil popülasyondaki en yüksek uygunluk değeri sahibi Ab'ler maksimum yanlış belirleme sayısı nedeniyle elenen Ab'lerin yerine birincil popülasyona kopyalanırlar; böylece Pop_p orjinal büyüklüğüne ulaşır. En sonunda, üzerinde-diğerlerinin-baskınlık-kurduğu (dominated) Ab'lerin hepsi Pop_s'den elenirler.

- Düzensiz (Ad-Hoc) Ağ oluşturma (opsiyonel): Pop_s'e kopyalanan yeni üstün-Ab'ler dağıtık ağda yayımlandıktan sonra, ağı dinlemekte olan jREMISA'lar tarafından kapılırlar ve eğer yeni gelen Ab kendi pop_s'indeki diğer Ab'lerden daha üstün ise kendi pop_s'lerine eklerler.

4.1.1.3 Deneyler ve Analizler

Deneylerde DARPA 1999 sızma belirleme verisetinin bir ve ikinci hafta verileri kullanılmıştır. Birinci hafta sadece kendinden-olan (normal) veri içermekte olup negatif seleksiyon aşamasında kullanılmıştır. İkinci haftanın %99.25'i kendinden-olan, %0.75'i kendinden-olmayan (etiketli saldırılar) veri içermekte olup öğrenme ve test aşamalarında kullanılmıştır. İkinci hafta verisetinin %99'u TCP/UDP/ICMP trafiğinden oluşmaktadır.

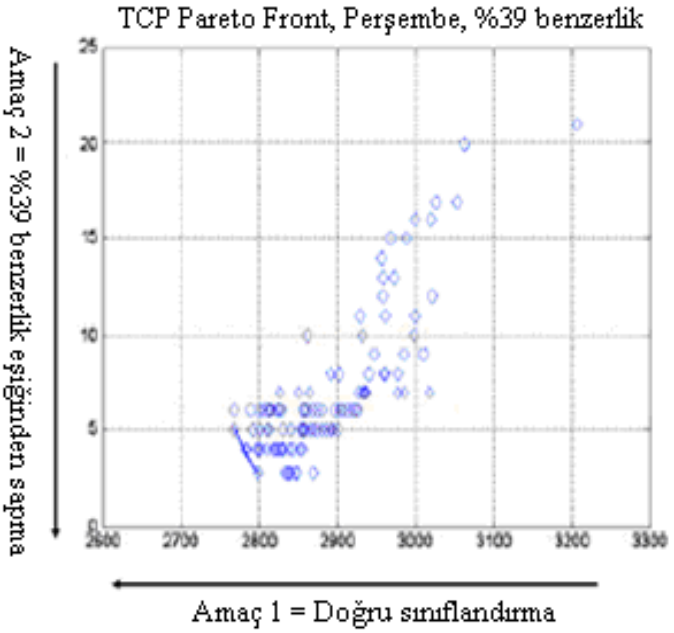
13 senaryonun 10 tanesinde, ikinci haftanın beş günü ayrı ayrı farklı eşik değerlerine göre kullanılmış olup, 3 tanesinde ise dağıtık ada modeli (distributed island) iki-, üç- ve dört-jREMISA düzeninde denenmiştir. 43 adet etiketli saldırılar arasından 16 tanesi değerlendirilmeye alınmıştır. Çizelge 4.1'de yer alan 1-6 arası senaryolarda ikinci hafta Perşembe günü verileriyle yapılan testlerde, her bir eşik değerine göre Ab'lerin ortalama etkinliği gösterilmektedir. Birleşik en yüksek sınıflandırma ve en düşük yanlış belirleme oranı 4'üncü senaryoda olduğu için diğer günlere ait senaryolarda (7-10) %39 eşik değeri kullanılmıştır.

Çizelge 4.1 : Tekli jREMISA test sonuçları [5].

								Kendinden-olan olaylar		Kendinden-olmayan olaylar	
Se-naryo	Gün	Nesiller	Benzerlik eşiği	TCP Pop	UDP Pop	ICMP Pop	Çalışma süresi	Doğru Neg%	Yanlış Neg%	Doğru Poz%	Yanlış Poz%
1	Perş.	1547710	42%	37	86	248	39.12 dk.	53.78	46.22	62.6	37.4
2	"	"	41%	106	116	284	52.48 dk.	67.44	32.56	68.33	31.67
3	"	"	40%	315	146	341	3.61 saat	76.10	23.90	76.92	23.08
4	"	"	39%	966	361	810	18.21 saat	85.45	14.55	97.66	2.34
5	"	"	38%	1580	423	881	2.36 gün	**86.48**	13.52	92.51	7.49
6	"	"	37%	2564	462	927	5.83 gün	82.52	17.48	99.71	0.29
7	Pzt.	1737455	39%	969	349	846	20.02 saat	85.36	14.64	**99.90**	0.10
8	Salı	1571748	"	922	362	882	18.86 saat	84.61	15.39	97.35	2.65
9	Çrşm.	995235	"	920	333	798	11.69 saat	83.37	16.63	98.26	1.74
10	Cuma	1347393	"	964	376	829	13.43 saat	83.59	16.41	96.57	3.43

Şekil 4.6'da, 4'üncü senaryonun TCP Pops çözümlerine ait grafik ve PF* gösterimi yer amaktadır. Çözümlerin %87'si + (%5) sapma alanında yer almıştır. Üstün

Ab'lerin dağıtık olarak diğer jREMISA'lar tarafından paylaşılması, tek jREMISA sistemlerinin etkinliğini yeteri kadar artıramamıştır.



Şekil 4.6 : TCP Pop_s PF* [5].

4.1.2 Geliştirilmiş-jREMISA

Daha iyi yanlış-pozitif ve yanlış-negatif değerler elde edebilmek için jREMISA üzerinde aşağıdaki geliştirmeler uygulanmıştır:

1. Negatif Seleksiyon, Uygunluk Fonksiyonu ve Klon Seleksiyon basamaklarındaki Hamming mesafesi değerlendirmeleri r-sürekli bit değerlendirmeleri kullanılarak iyileştirildi. Karşılaştırılan iki kromozomun benzer olarak kabul edilebilmesi için en az "r" sayısınca ardışık bitlerinin aynı olması gerekmektedir. Daha güçlü bir değerlendirme elde etmek için r-sürekli bit gereksinimi, önceki Hamming mesafesi hesaplamaları ile beraber kullanıldı.
2. Kendinden-olan Ag'leri tanıyan Negatif Seleksiyon'un rastgele Ab'leri elenmekte; fakat olgun (elenmeyen) Ab'ler önceden belirlenmiş olan popülasyon büyüklüğüne ulaşana kadar elenen Ab'lerin yerine yeni rastgele Ab'ler yaratılmaktadır. Böylece, her şartta yeterli sayıda olgun Ab olacaktır.

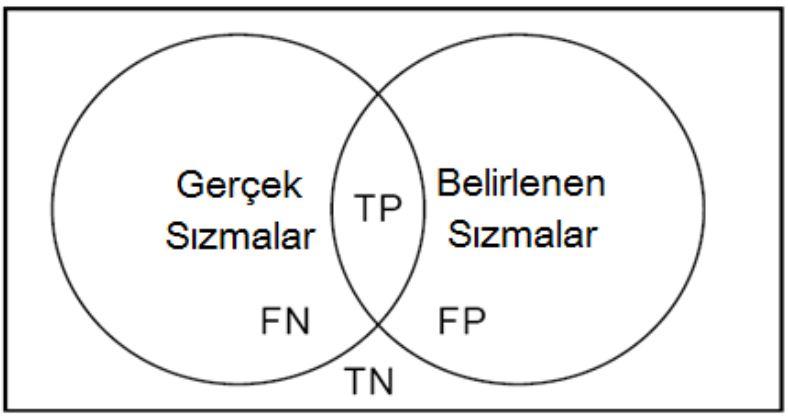
3. İkinci hedef olarak, benzerlik eşiğinden olan sapmanın minimize edilmesi yerine yanlış pozitif ve negatif skorların minimize edilmesi kullanıldı.
4. Klon seleksiyon sonunda ikincil popülasyondaki üzerinde-diğerlerinin-baskınlık-kurduğu Ab'lerin tamamı elenmemekte; fakat ikincil popülasyonun büyüklüğü birincil popülasyonun büyüklüğüne eşit olacak kadar kısaltılmakta ve geri kalan üzerinde-diğerlerinin-baskınlık-kurduğu Ab'ler baskınlık değerlerine göre sıralanmaktadırlar.
5. Elit olma özelliklerini bozmamak için klonlanan Ab'lerin orjinallerine Klon Seleksiyonda Uniform Mutasyon uygulanmadı.

Yukarıda anlatılan eklemeler, orjinal jREMISA gibi sadece iyi eğitim sonuçları değil aynı zamanda iyi test sonuçları alabilmek için yapıldı. R-sürekli bit değerlendirmesi, yanlış belirlemeleri azaltmaktadır çünkü Ab benzerlikleri için başka bir zorlayıcı kriter daha eklemektedir. İki numaralı geliştirme öncesinde, Negatif Seleksiyon işlemi sonrasında 42 ve üzeri yüksek eşik değerleri altında yeterli sayıda kendinden-olmayan Ab popülasyonu elde edilemiyordu. Benzerlik eşiğinden olan sapmanın minimize edilmesinden vazgeçildi çünkü üç numaralı geliştirmede de bahsedildiği gibi sistem büyük sapma değerlerinde daha iyi sonuçlar vermektedir.

4.2 Uygulama ve Deneyler

SBS'nin temel amacı bilgisayar sistemlerinin iç ve dış saldırıcılar tarafından yanlış, yetkisiz ve zararlı kullanımını belirlemektir. Kritik olan nokta, doğrulanmamış uyarıların (yanlış-pozitif) oluşumunu minimize ederken doğru uyarıları (doğru pozitif) maksimize etmektir. SBS'lerin değerlendirilmesinde yaygın olarak kullanılan ölçütler Şekil 4.7'de gösterilmiştir:

- Doğru pozitif (TP): doğru bir şekilde saldırı olarak sınıflandırılan gerçek saldırı,
- Yanlış pozitif (FP): normal veri için yanlışlıkla verilen yanlış alarm,
- Doğru Negatif (TN): doğru bir şekilde uyarıya neden olmayan normal veri,
- False negative (FN): yanlışlıkla normal olarak sınıflandırılan bir saldırı.



Şekil 4.7 : Değerlendirme ölçütleri.

İlk grup testlerde, sadece DDoS saldırısına özel olan MIT DARPA 2000 LLS_DDOS 1.0 veriseti kullanılarak daha iyi doğru ve yanlış pozitif sonuçlar elde edecek en iyi parametre grubunu bulabilmek hedeflenmiştir. Fakat literatürde DARPA verisetinin bu sürümünü kullanarak sonuçlar veren çok çalışma bulunmamaktadır. Geliştirilmiş-jREMISA'nın performansını diğer benzer çalışmalarla karşılaştırabilmek için literatürde yaygın olarak kullanılan MIT DARPA 1999 SBS verisetini kullanarak ikinci grup testler yapıldı.

4.2.1 Uygulama-1

4.2.1.1 Test tasarımları

Testler, Windows XP SP3 ile çalışan Pentium Core 2 Duo 2.4 GHz bilgisayarlar ile gerçekleştirilmiştir. Geliştirilmiş-jREMISA'da, daha iyi doğru ve yanlış pozitif sonuçlar elde edecek en iyi parametre grubunu bulabilmek için benzerlik eşiği, r-sürekli değerler ve birincil popülasyon büyüklüğü parametrelerinin farklı düzenlemelerinin yapıldığı üç farklı test yer almaktadır. Son olarak, orjinal ve geliştirilmiş-jREMISA, aralarındaki gelişimi görebilmek için değişen eşik değerlerine göre karşılaştırıldılar. Gerçekten belirlenen tüm saldırıların oranı olan doğru pozitif oranı ve (yanlış) uyarı verdiren normal verilerin oranı olan yanlış pozitif oranından oluşan değerlendirme ölçütlerinin ortalama ve standart sapmalarını bulabilmek için her bir test 20'şer kez çalıştırıldı.

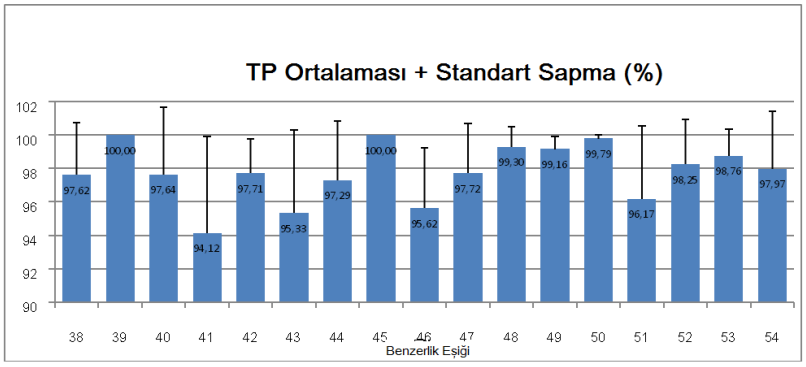
4.2.1.2 Kullanılan veriseti

Birinci grup uygulamalarda MIT DARPA 2000 LLS_DDOS 1.0 veriseti kullanıldı. Ne yazık ki, bu verisetinin doğruluk-kümesi Lincoln Labaratuarı web sayfasında

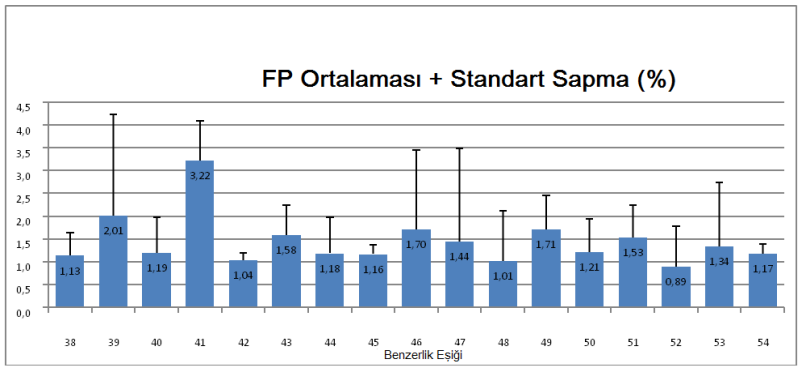
verilmemektedir. Doğruluk-kümesi, saldırının yapısı ve saldırıcının kimliği dikkate alınarak oluşturuldu. Ethereal [72] vasıtasıyla DDoS saldırısıyla ilişkili gelen-trafiğin tamamı kaldırılarak Ab popülasyonunun eğitilebileceği, sadece kendinden-olan bir veriseti oluşturuldu. Saldırı trafiği içeren orjinal verisetinin olduğu testlerde Negatif Seleksiyon ve ÇAEA sonrası oluşan başarılı ikincil popülasyon Ab kromozomları kullanıldı.

4.2.1.3 Sonuçlar

Benzerlik eşiği testleri. En iyi doğru ve yanlış pozitifleri veren en iyi eşik değerine karar verebilmek amacıyla geliştirilmiş-jREMISA üzerine %38 - %54 aralığında değişen benzerlik eşiği değerleri uygulanmıştır. Bu eşik değerleri, orjinal jREMISA çalışmasındaki deneyler dikkate alınarak seçilmiştir. TCP, UDP ve ICMP popülasyonlarının her birinin büyüklüğü 100 kromozomdur ve r-sürekli değer olarak 10 sayısı kullanılmıştır. Sistem %54 üzerindeki eşik değerleri için çalışmamaktadır. Şekil 4.8 ve Şekil 4.9'da görüldüğü gibi, sonuçlar arasında fazla fark yoktur; doğru pozitif oranlarının çoğu %97 üzerinde (bazıları %100), yanlış pozitif oranlarının çoğu %1,5'un altındadır.

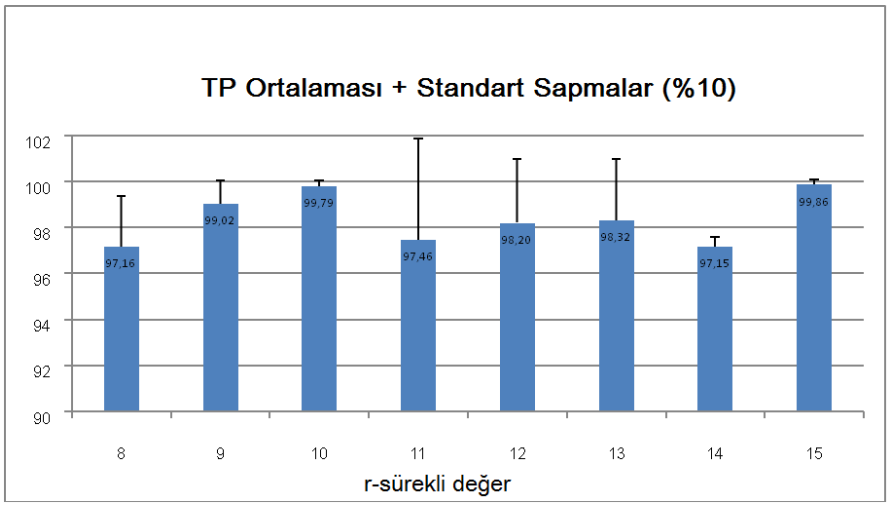


Şekil 4.8 : Değişen benzerlik eşiklerinin doğru pozitif oranları (%).

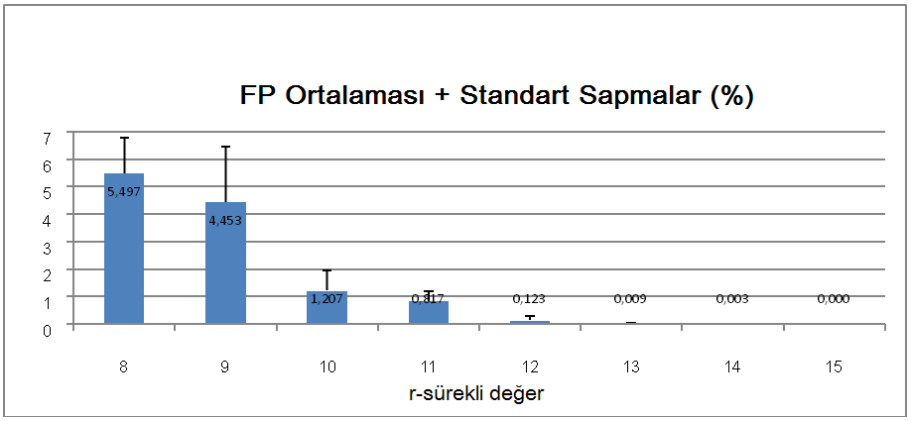


Şekil 4.9 : Değişen benzerlik eşiklerinin yanlış pozitif oranları (%).

R-sürekli testleri. En iyi doğru ve yanlış pozitifleri veren en iyi r-sürekli değerlerine karar verebilmek amacıyla geliştirilmiş-jREMISA üzerine 8-15 aralığında değişen r-sürekli değerleri uygulanmıştır. TCP, UDP ve ICMP popülasyonlarının her birinin büyüklüğü 100 kromozomdur ve benzerlik eşik değeri olarak %50 oranı kullanılmıştır. Şekil 4.10 ve Şekil 4.11'de görüldüğü gibi doğru pozitiflerin sonuçları arasında büyük farklar yoktur; çoğu %97'nin üzerindedir. Ilk ikisi dışında yanlış pozitif oranları %1,5'un altındadır. R-sürekli değeri 15 olduğunda %0 yanlış pozitif oranı gerçekleşmektedir.



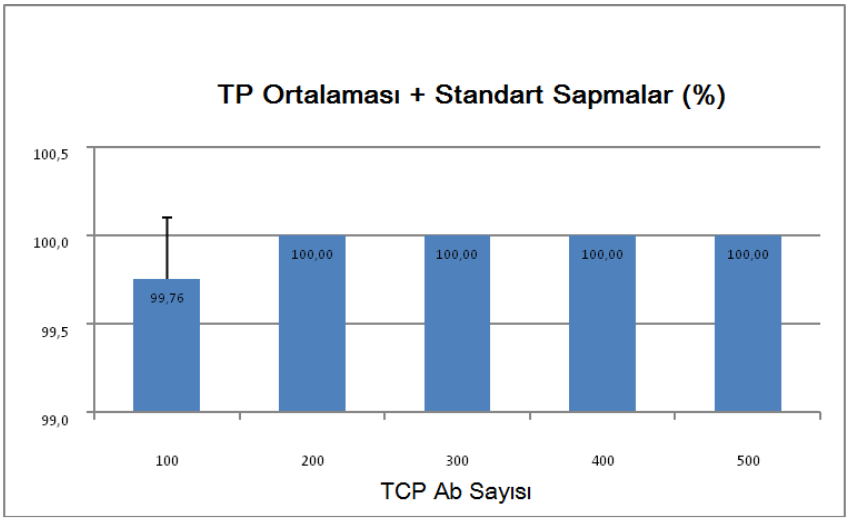
Şekil 4.10 : Değişen r-sürekli değerlerine göre doğru pozitif oranları (%).



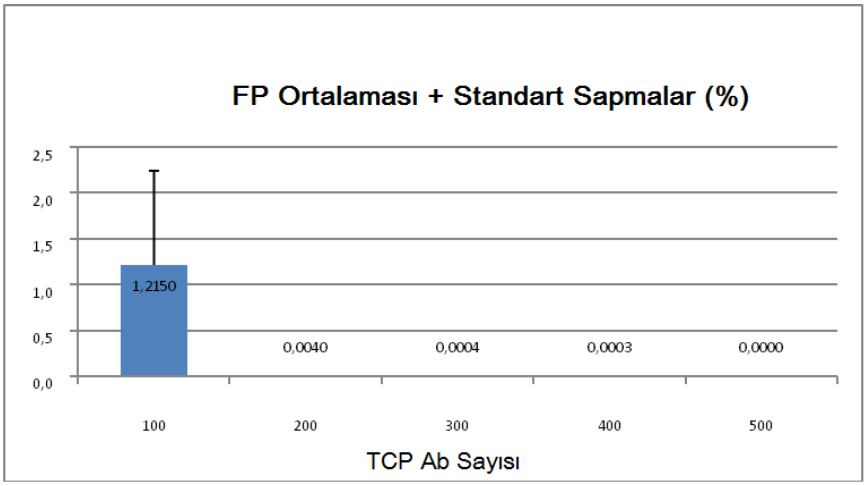
Şekil 4.11 : Değişen r-sürekli değerlerine göre yanlış pozitif oranları (%).

Pop_p Büyüklük Testleri. En iyi doğru ve yanlış pozitifleri veren en iyi TCP Ab popülasyon büyüklüğüne karar verebilmek amacıyla, geliştirilmiş-jREMISA üzerine UDP ve ICMP Ab popülasyon büyüklükleri sabit alınarak 100-500 aralığında değişen TCP Ab büyüklükleri uygulanmıştır. Benzerlik eşik değeri olarak %50, r-sürekli değer olarak 10 sayısı kullanılmıştır.

Şekil 4.12'de görüldüğü gibi, doğru pozitiflerin sonuçları arasında fazla fark bulunmamaktadır; birincisi dışındaki değerlerin hepsi %100'dür. Şekil 4.13, birincisi dışındaki yanlış pozitif oranların %0.1'in altında olduğunu göstermektedir. SBS'in eğitim bölümü çevrim-dışı olarak icra edileceği için popülasyon büyüklüğünü artırmanın neden olacağı işlemci yükü göz ardı edilmiştir.



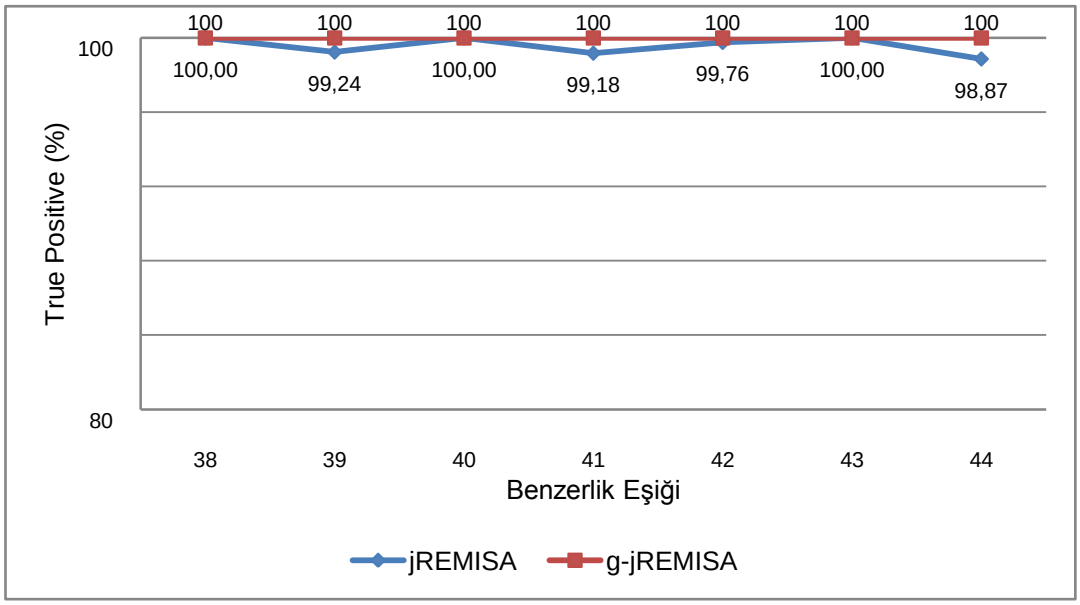
Şekil 4.12 : Değişen Pop_p TCP Ab sayılarına göre doğru pozitif oranları (%).



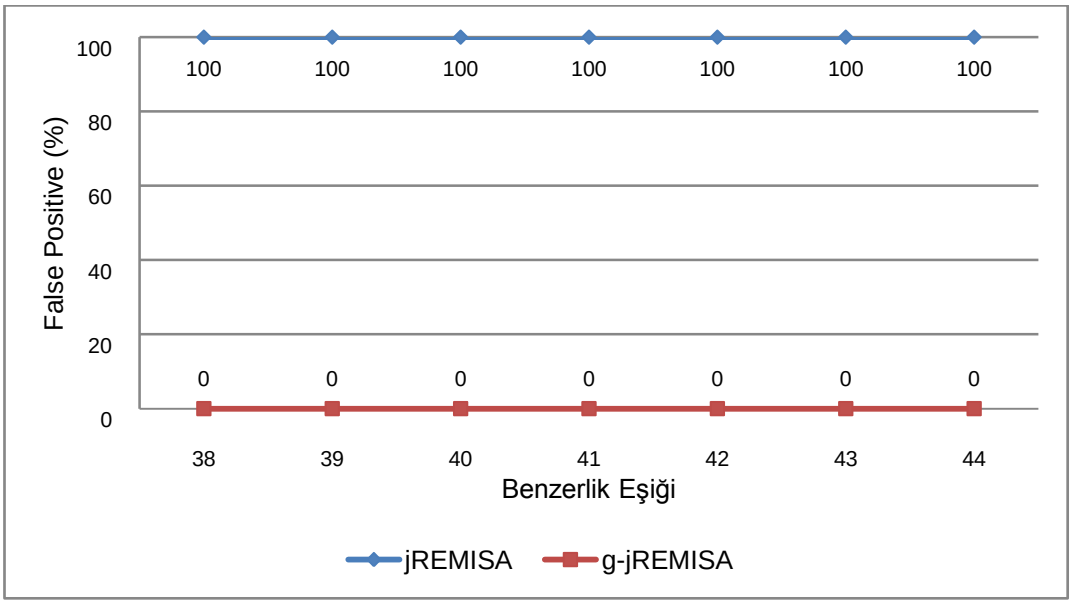
Şekil 4.13 : Değişen Pop_p TCP Ab sayılarına göre yanlış pozitif oranları (%).

Orjinal ve geliştirilmiş jREMISA'nın karşılaştırılması. Orjinal ve geliştirilmiş jREMISA'nın MIT DARPA 2000 LLS_DDOS 1.0 veriseti üzerindeki performansı karşılaştırıldığında, geliştirilmiş versiyonun orjinalinden çok daha iyi olduğu görülmektedir. TCP, UDP ve ICMP popülasyon büyüklükleri sırasıyla 300,100 ve 100; r-sürekli değerin 10 olduğu durumda geliştirilmiş-jREMISA'nın ve orjinal jREMISA'nın tüm doğru pozitif oranları Şekil 4.14'de görüldüğü gibi yaklaşık

olarak %100'dür. Fakat, yanlış pozitif oranlarındaki büyük fark Şekil 4.15'de görülebilmektedir. Geliştirilmiş-jREMISA'nın yanlış pozitif oranlarının hepsi %0 iken, orjinal jREMISA'nın yanlış pozitif oranlarının hepsi %100'dür. Orjinal jREMISA, eğitim aşamasında aldığı başarılı ara sonuçları test aşamasında gösterememektedir.



Şekil 4.14 : Değişen eşik değerlerine göre doğru pozitiflerin karşılaştırılması (%).



Şekil 4.15 : Değişen eşik değerlerine göre yanlış pozitiflerin karşılaştırılması (%).

4.2.2 Uygulama-2

Önceki uygulamada DARPA verisetinin 2000 sürümü kullanılarak jREMISA[5] üzerine yapılan geliştirmelerin DDoS saldırıları üzerindeki etkileri incelendi. Fakat,

literatürde DDoS saldırılarını belirlemek için verisetinin bu sürümünü kullanarak sonuçlar veren bir çalışma bulunmamaktadır. Geliştirilmiş-jREMISA'nın performansını diğer benzer çalışmalarla karşılaştırabilmek için 1999 DARPA SBS veriseti günlerinin farklı bileşimleri kullanılarak iki çeşit deney yapıldı.

4.2.2.1 Test tasarımları

Deneyler, Windows XP SP3 işletim sistemi ile çalışan Pentium Core 2 Duo 2.4 GHz bilgisayarlarda gerçekleştirildi. İlk deney tipinde sistem, birinci haftanın farklı günlerindeki saldırı-içermeyen verisetleriyle eğitilip, dördüncü haftanın aynı adlı günlerindeki saldırı-içeren verisetleriyle test edildi. Ama ikinci tipte, sistemin daha gerçekçi durumlardaki performansını ölçmek için test aşamasında farklı adlı günlerdeki verisetleri kullanıldı. Her deney, doğru pozitif ve yanlış pozitif oranlarının oluşturduğu değerlendirme ölçütlerinin ortalama değerlerini bulabilmek için 20'şer kez koşturuldu.

R-sürekli değer için 15, benzerlik eşiği için %50 ve TCP, UDP, ICMP popülasyonlarının büyüklükleri için sırasıyla 300, 100, 100 değerleri kullanıldı. Bu değerler, DARPA LLS_DDOS 1.0 veriseti üzerinde yapılan deneylerde en iyi parametre ayarları olarak bulunmuştur. Negatif Seleksiyon eğitiminde saldırı-içermeyen hafta-1 verisetinin beş günü kullanılmış olup, ÇAEA eğitimi ve testlerinde hafta-4'ün dört günü ve hafta-5'in Salı günü (hafta-4'ün Salı günü verilmemiştir) kullanılmıştır. Saldırı paketlerini içeren doğruluk-kümesi (truthset), Ethereal yazılımı ve Lincoln Laboratuarı web sitesinde verilen tanımlama listeleri yardımıyla oluşturuldu. ÇAEA aşamasının sonunda en iyi sonuçlar olarak elde edilen ikincil popülasyon Ab kromozomları, saldırı trafiği içeren veriseti üzerinde yapılan test aşamasında kullanıldılar.

4.2.2.2 Kullanılan veriseti

DARPA 1999 değerlendirmelerinin yaklaşık üç ay aralıklı olarak iki aşaması vardır. İlk aşama sırasında, katılımcılara üç haftalık veri sağlanmıştır. Birinci ve üçüncü haftalar saldırı içermemektedir ve anomali belirleme sistemlerini eğitmek için kullanılabilirler.

İkinci aşama sırasında, katılımcılara 40 tanesi eğitim verisinde yer almayan 58 saldırının 201 örneğini içeren iki haftalık (4 ve 5'inci haftalar) test verisi sağlanmıştır [73]. Saldırıların sınıflandırması aşağıda yer almaktadır:

- Yoklama, tarama veya keşif saldırıları: Bir bilgisayar ağını veya bir DNS sunucuyu, geçerli IP adreslerini, aktif portları, konakçılardaki işletim sistemlerini ve bilinen zayıflıkları bulmak için otomatik olarak tarayan saldırılardır. Keşif saldırıları genelde bir saldırının ilk basamağıdır; eğer kullanışlı bilgi bulunursa başka tipte bir saldırıyla devam edilir.
- Uzaktan-yerele (R2L) saldırıları: Bu saldırılarda, hedef makine üzerinde bir kullanıcı hesabı olmayan bir saldırıcı, makineye yerel erişim sağlar, dosyaları makineden alır veya verileri değiştirir.
- Kullanıcıdan-yöneticiye (U2R) veya yetki yükseltme saldırıları: Bir makine üzerindeki yerel bir yetkisiz kullanıcının, UNIX super kullanıcısı veya Windows NT yöneticisi için olan yetkileri elde edebildiği saldırılardır.
- Hizmetin engellenmesi saldırısı (DoS): Bu sınıftaki saldırılar, bir konakçı veya ağ hizmetini uzaktan kapatmaya veya bozmaya çalışırlar. DDoS saldırıları, aynı şeyi çok sayıda kaynak kullanarak yapmaya çalışır.

4.2.2.3 Sonuçlar

İlk deney kümesinde, Çizelge 4.2'de görüldüğü gibi eğitim ve test aşamalarında hafta-1 ve hafta-4'ün aynı günleri kullanılmıştır. Örneğin, eğer eğitim safhasının Negatif Seleksiyon ve MOEA aşamalarında hafta-1'in Çarşamba'sı kullanıldıysa, testlerde hafta-4'ün Çarşamba'sı kullanıldı. Bu iki veriseti, sadece hafta-1 saldırı-içermeyen ve hafta-4 saldırı-içeren olduğu için değil, aynı zamanda kullanılan protokollerin miktarı farklı olduğu için de birbirlerinden farklıdırlar.

Örneğin, hafta-1'in Pazartesi'sinde % 92,67 TCP, %7,26 UDP and %0,07 ICMP trafik vardır; fakat hafta-4'ün Pazartesi'sinde %79,08 TCP, %17,72 UDP and %3,20 ICMP trafik bulunmaktadır. Her bir günün 5 gün üzerinden ortalaması alınınca, Doğru Pozitif oranları yaklaşık %100 ve Yanlış Pozitif oranları yaklaşık %0 çıkmaktadır.

Çizelge 4.2 : İlk deney kümesinin sonuçları.

Eğitim		Test (hafta-4)	Doğru Pozitif (%)	Yanlış Pozitif (%)
Negatif Seleksiyon (hafta-1)	MOEA (hafta-4)			
Pazartesi	Pazartesi	Pazartesi	100	0
Salı	Salı	Salı	99,15	0
Çarşamba	Çarşamba	Çarşamba	100	0
Perşembe	Perşembe	Perşembe	99,83	0
Cuma	Cuma	Cuma	100	0
Ortalama			99,8	0

Orjinal jREMISA'nın DARPA 1999 verisetine verdiği TP/FP cevaplarının LLDDOS 1.0 verisetine verdiği cevaplardan farklı olmadığı test edilerek gözlemlenmiştir. Geliştirilmiş-jREMISA'nın özellikle FP oranlarında sağladığı iyileştirme Çizelge 4.3 incelenerek anlaşılmaktadır.

Çizelge 4.3 : Orjinal jREMISA DARPA99 sonuçları.

Eğitim		Test (hafta-4)	Doğru Pozitif (%)	Yanlış Pozitif (%)
Negatif Seleksiyon (hafta-1)	MOEA (hafta-4)			
Pazartesi	Pazartesi	Pazartesi	100	100
Salı	Salı	Salı	100	100
Çarşamba	Çarşamba	Çarşamba	99,08	100
Perşembe	Perşembe	Perşembe	98,56	100
Cuma	Cuma	Cuma	100	100
Ortalama			99,7	100

İkinci deney kümesi, eğitim ve test aşamalarında hafta-4'ün farklı günlerini kullanarak önceki test kümesinden farklılaşmaktadır. Yanlış Pozitif ve Doğru Pozitif oranları sırasıyla %0 ve %100'den fazla sapmamaktadır. Sistemimiz, Çizelge 4.4'de görüldüğü gibi, daha gerçekçi olan bu test durumlarında da iyi sonuçlar vermektedir.

Çizelge 4.4 : İkinci-tip deneylerin sonuçları.

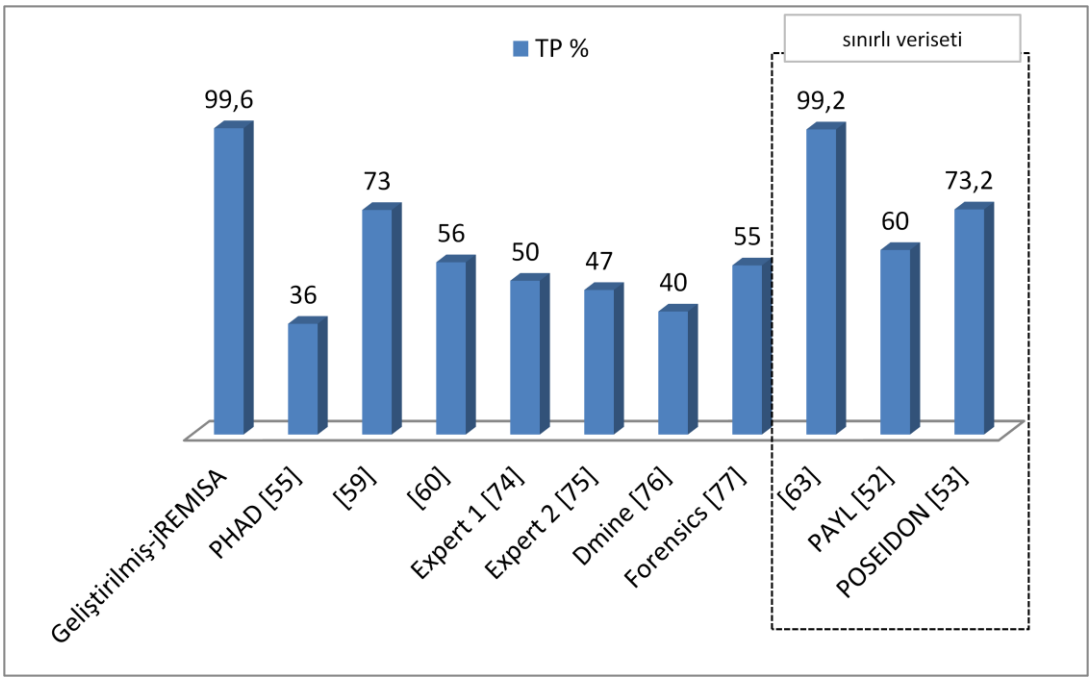
Eğitim		Test (hafta-4)	Doğru Pozitif (%)	Yanlış Pozitif (%)
Negatif Seleksiyon (hafta-1)	MOEA (hafta-4)			
Beş günün bileşimi	Pazartesi	Salı*	100	0,03
	Pazartesi	Çarşamba	99,17	0,08
	Pazartesi	Perşembe	99,45	0,06
	Pazartesi	Cuma	100	0,005
	Perşembe	Pazartesi	99,63	0,002
	Perşembe	Cuma	100	0,007
Ortalama			99,69	0,0307

jREMISA çalışması üzerinde yapılan beş ana düzeltmenin hangilerinin geliştirilmiş-jREMISA sonuçlarını ne kadar etkilediğinin ölçülmesi amacıyla yapılan testlerin sonuçları Çizelge 4.5'de gösterilmiştir. Testlerde, düzeltmeler teker teker eksik bırakılarak g-jREMISA'nın farklı verisetlerine verdiği TP ve FP cevapları elde edilmiştir. İkinci düzeltme eksiltildiğinde sistem, r-süreklilik olmaksızın 38 eşik değeri ile çalıştırılmıştır. Bu sonuçların tüm düzeltmelerin var olduğu g-jREMISA sonuçlarından ne kadar farklı oldukları çizelgede incelenebilmektedir. En büyük etkiyi FP oranlarını dahi fazlaca değiştiren birinci düzeltme olan "r-süreklilik kısıtının eklenmesi" düzeltmesinin yaptığı, üçüncü düzeltme olan "ikinci hedefin değiştirilmesi" düzeltmesinin de sadece TP oranları üzerinde belirgin değişikliklere neden olduğu gözlemlenmiştir.

Çizelge 4.5: g-jREMISA düzeltmelerinin etkileri.

NEG.SEL.	**MOEA**	**TEST**	**%**	**TAM ÇLŞ.**	**EKSİK OLAN DÜZELTMELER**				
					D1	**D2**	**D3**	**D4**	**D5**
LLDDOS-1	LLDDOS-1	LLDDOS-1	**TP**	100	94.23	100	94.16	100	100
			FP	0	99.9	0.23	0.005	0.0005	0.003
99H1	99H4_PZT	99H4_ÇRŞ	**TP**	99,17	94.55	96.9	91	97	95.29
			FP	0.08	71.7	0.14	0.001	0.3	0.01
99H1	99H4_PZT	99H4_PRŞ	**TP**	99,45	92.5	97.5	91.98	97.9	95.87
			FP	0.06	68.25	0.04	0.0027	0.0035	0.016
99H1	99H4_PZT	99H5_SALI	**TP**	100	90.45	94.79	88.21	95.88	92.23
			FP	0.03	77.08	0.007	0.001	0.045	0.008

Geliştirilmiş-jREMISA'nın birinci-tip deneylerdeki performansı ve diğer benzer çalışmaların DARPA 1999 veriseti üzerindeki sonuçları karşılaştırıldığında, geliştirilmiş-jREMISA'nın diğerlerinden daha iyi TP ve FP oranlarına sahip oldukları görülmüştür. Bu karşılaştırmanın yapıldığı Şekil 4.16 ve Çizelge 4.6'nın son üç çalışmasında, sadece TCP trafiğinden oluşan sınırlı veriseti kullanılmıştır. 2.2.2 numaralı bölümde değerleri verilen çalışmalar hakkında detaylı bilgi verilmektedir.



Şekil 4.16 : Geliştirilmiş-jREMISA'nın diğer çalışmalarla TP karşılaştırması.

Çizelge 4.6 : Sistemlerin FP oranları.

Sistemler	Geliştirilmiş -jREMISA	[55]	[59]	[60]	[74]	[75]	[76]	[77]	[63]	[52]	[53]
FP değerler	0,03%	10 FP/ gün	Yok	Yok	10 FP/ gün	10 FP/ gün	10 FP/ gün	10 FP/ gün	4%	1%	1%

5. DDOS-ÖNCESİ DAĞITIK SALDIRILARI BELİRLEYEN ANOMALİ-TEMELLİ ADAPTİF SIZMA BELİRLEME SİSTEMİ

Çalışmanın üçüncü bölümde ele alınan "gezgin etmenler kullanan dağıtık imza-temelli SBS" aşamasındaki dağıtık yapı ile dördüncü bölümde anlatılan "yapay bağışıklık sisteminden esinlenen çok-amaçlı evrimsel algoritma kullanan anomali-temelli SBS" aşamasındaki doğadan esinlenen algoritmalar tarafından akıllı hale getirilmiş olan anomali-temelli SBS birleştirilmiş ve sonradan eklenecek saldırı tiplerine de kolay adapte olabilme özelliği eklenerek hedeflenen sistem oluşturulmuştur. Önceki aşamalarda elde edilen sonuçlar ve bu aşamada yapılan test sonuçları değerlendirilerek hızlı, yüksek güvenirlilik sahibi ve ağda optimum düzeyde yüke yol açan modüler bir yöntem önerilmiştir. Bu bölümde, önerilen yöntem ve onu oluşturan iki alt yöntem ile MIT DARPA 2000 LLS_DDOS 1.0 ve 2.0.2 verisetleri kullanılarak yapılan testler ve sonuçları açıklanmış olup sisteme kazandırılan adaptif yaklaşım ayrıca ele alınmıştır.

5.1 Sistem Tasarımı

Üçüncü bölümde anlatılan imza-temelli dağıtık SBS yöntemlerinde, her bir konakçıda çalışan Static Agent'lar kısa periyotlarla kendi konakçılarında yer alan ve önceden SNORT tarafından oluşturulmuş olan "alarm.ids" dosyasındaki saldırı-şüphesi kayıtlarını incelemekte ve bu saldırı-şüphelerine SNORT tarafından verilen isim ile beraber diğer bazı bilgileri ya MainAgent'a mesajlamakta veya saldırıya özgün yaratılan Mobile Agent'a vermektedir. Dağıtık bir saldırı kararı verebilmek için başka bir konakçıdan belirli bir süre içerisinde gelen aynı isimli saldırı-şüphesi bilgisinin kaynak-IP değeri kontrol edilerek saldırı trafiğinin aynı kaynak tarafından gönderilip gönderilmediği incelenmektedir. SNORT, saldırı-şüphesi kayıtlarını oluştururken, paketlerin sadece "header" bilgilerini kontrol etmekte, "dataload" bilgileriyle ilgilenmemektedir.

Çalışmanın imza-temelli ilk aşamasında paketlerin "header" bilgileri SNORT kural veritabanı ile eşleştirilerek saldırı kararı verilmektedir; bu nedenle MIT DARPA LLS_DDOS 1.0 verisetinin dördüncü aşamasında gerçekleşen "upload" trafiği yapılan testlerde tespit edilememiştir.

Çalışmanın bu aşamasında, henüz sisteme tanıtılmamış, ilk defa geliştirilen saldırı trafiklerini ve "upload" paketlerini tespit edebilmek amacıyla paketlerin "dataload" kısmını da inceleyen anomali-temelli bir dağıtık SBS kurulmuş ve yapılan testlerin sonuçları irdelenmiştir. İmza-temelli SBS çalıştırılan ilk aşamada her bir konakçıda yer alan ve çevrim-dışı olarak SNORT trafından yaratılan saldırı-şüphesi kayıtlarının yerini g-jREMISA tarafından yine çevrim-dışı olarak yaratılan kayıtlar almıştır. Şekil 5.1'de kısa bir örneği gösterilen imza-temelli SBS ürünü kayıtlarda paketlerin "dataload" kısmı bulunmazken, Şekil 5.2'de kısa bir örneği gösterilen anomali-temelli SBS ürünü kayıtlarda "dataload" bölümü de bulunmaktadır. Her iki örnek de karşılaştırma yapılabilmesi için MIT DARPA 2000 LLS_DDOS 1.0 verisetinden alınmıştır. Aslında tek bir satırdan oluşan bir paket kaydı şekil içerisine sığdırmak için birden fazla satırda ifade edilmiştir. Şekil 5.2'deki her bir kayıttta sırasıyla "header" bölümündeki bilgilerden derlenen protocol kodu (1= TCP, 2=UDP, 3=ICMP), paket numarası, tarih, saat, tarih ve saatin UNIX EPOCH olarak ifadesi, kaynak-IP, hedef-IP, sızma tipini temsil eden kromozom dizilişi ve sonrasında "dataload" bölümünde yer alan veriler gösterilmiştir.

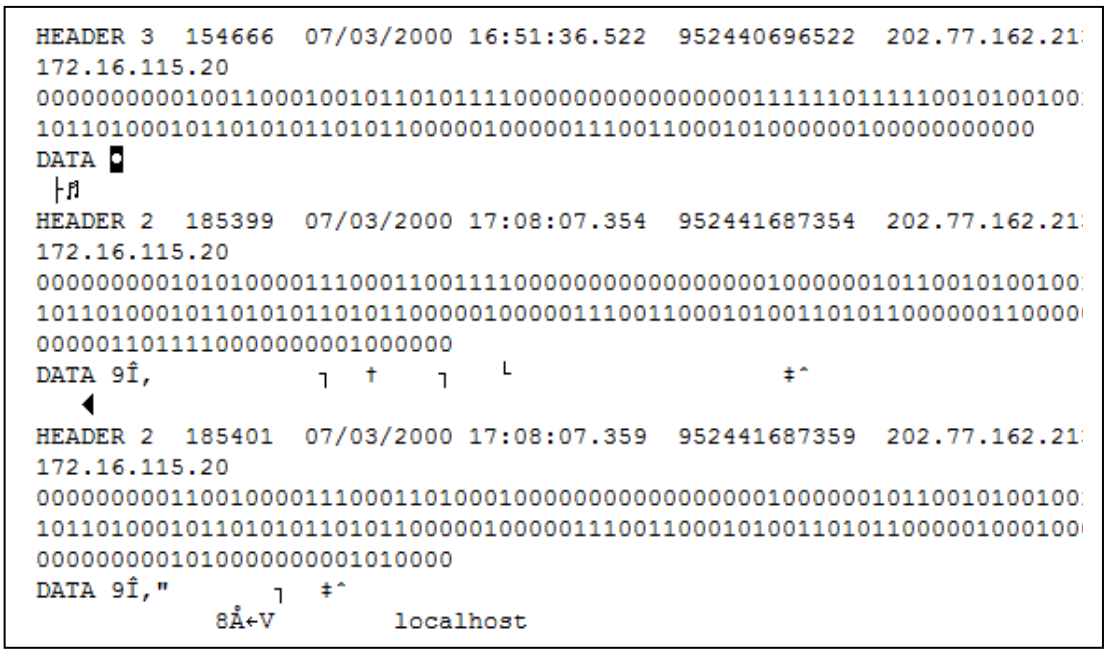
```
03/07-16:32:34.019149  [**] [1:491:8] INFO FTP Bad login [**]
[Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
172.16.115.20:21 -> 172.16.113.50:1045

03/07-16:51:36.522920  [**] [1:384:5] ICMP PING [**]
[Classification: Misc activity] [Priority: 3] {ICMP}
202.77.162.213 -> 172.16.115.20

03/07-17:08:07.354091  [**] [1:585:7] RPC portmap sadmind request
UDP [**] [Classification: Decode of an RPC Query] [Priority: 2]
{UDP} 202.77.162.213:54790 -> 172.16.115.20:111

03/07-17:08:07.359636  [**] [1:1957:7] RPC sadmind UDP PING [**]
[Classification: Attempted Administrator Privilege Gain]
[Priority: 1] {UDP} 202.77.162.213:54792 -> 172.16.115.20:32773
```

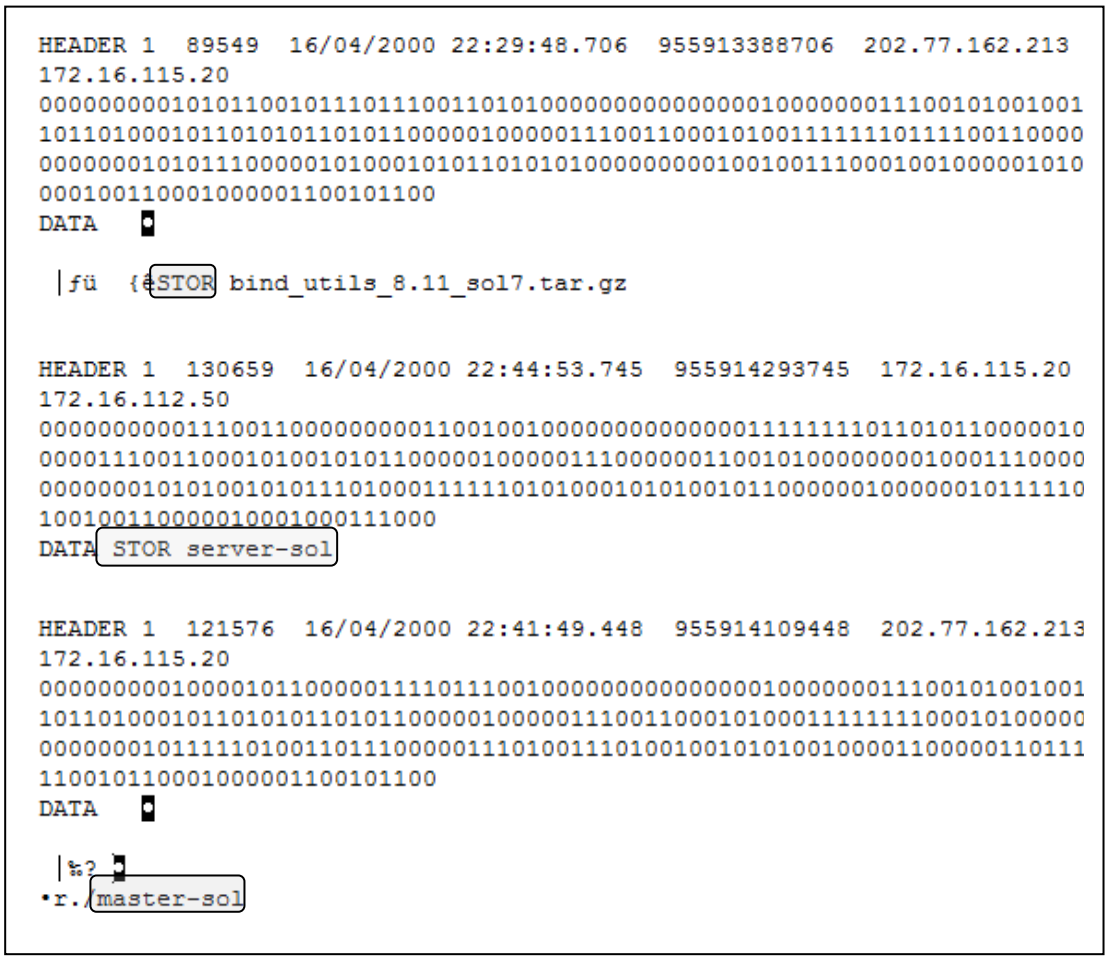
Şekil 5.1 : İmza-temelli SBS ürünü olan "alarm.ids" kayıtları.



Şekil 5.2 : Anomali-temelli SBS ürünü olan "alarm.ids" kayıtları.

Sızma tiplerini temsil eden Ab kromozomları birbirleriyle karşılaştırılmakta ve Hamming mesafesi değeri kromozom uzunluğunun % 80'inden büyük ve r-sürekli değeri 15'den büyük olduğunda paketlerin birbirlerine benzediğine karar verilerek bir saat içerisinde aynı kaynak-IP'den başlatılıp başlatılmadığı kontrol edilmektedir. Bu kontrol de olumlu sonuçlanırsa dağıtık bir saldırı yapıldığı düşüncesiyle güvenlik konsolünde uyarı mesajı verilmektedir. Belirlenen ilk dağıtık saldırıya DDOS-1 ismi verilmekte ve her yeni dağıtık saldırıda bu rakam artırılmaktadır. Güvenlik konsolunda saldırılar hakkında gerekli bilgiler verilerek, güvenlik yöneticisinin ilgili işlemleri yapmasına imkan tanınmaktadır.

DDoS saldırılarında saldırıcı, saldırıları başlatmak için kullanacağı orta katmandaki konakçılara mutlaka bir "upload" işlemi sonrasında "trojan" işlevinde bir program kurar. Bu konakçılardan bir tanesini, diğerlerini yönetmek için "master", diğerlerini "server" olarak belirler. Bu çalışmada geliştirilen sistem de, aynı kaynak tarafından dağıtık bir "upload" işlemi yapılıp yapılmadığını, paketlerin "dataload" bölümlerine bakarak kontrol etmekte ve güvenlik yöneticisini haberdar etmektedir. Şekil 5.3'de farklı zamanlarda gerçekleşen "upload" işlemine ait paketlerin görüntüsü gösterilmektedir. "STOR", "server-sol" ve "master-sol" kelimeleri anahtar kelimelerdir.



Şekil 5.3 : "upload" trafiklerine ait görüntüler.

Dağıtık bir saldırı şüphesi sonrası dağıtık bir "upload" yapılmış olması, bir DDoS saldırısı başlatılacağının en kuvvetli işareti olacağından güvenlik konsolunda "DDOS ALARM" uyarısı verilerek güvenlik yöneticisi haberdar edilmektedir. Şekil 5.4'de güvenlik konsolunun bir görüntüsü yer almaktadır. Sırasıyla, saldırı adı, protokol kodu, kaynak-IP, saldırıya uğrayan ilk konakçı IP adresi, saldırıya uğrayan ikinci konakçı IP adresi, saldırının ikinci konakçıda gerçekleşme zamanı bilgileri de verilmektedir.

```
DDOS-1 3 202.77.162.213 172.16.115.5 172.16.115.20 07/03/2000 16:51:36.522
DDOS-2 2 202.77.162.213 172.16.115.20 172.16.115.87 07/03/2000 17:08:07.495
DDOS-3 1 202.77.162.213 172.16.115.20 172.16.112.10 07/03/2000 17:34:40.489
DDOS-4 1 202.77.162.213 172.16.115.20 172.16.112.10 07/03/2000 17:50:21.296
UPLOAD 202.77.162.213 172.16.115.20 172.16.112.10 07/03/2000 17:50:21.296
!!!!!!!DDOS ALARM!!!!!!!DDOS ALARM!!!!!!!!DDOS ALARM!!!!!!!!
```

Şekil 5.4 : Güvenlik konsolu görüntüsü.

Çalışmanın son aşamasında yapılan düzenlemeler neticesinde önceden tasarlanan Yöntem_2, Yöntem_3 ve Yöntem_4 yaklaşımlarının kullanılmasına gerek kalmadığı ortaya çıktı. Yöntem_1, "Merkezi Yöntem"; Yöntem_5, "Tam-dağıtık Yöntem"; Yöntem_6 ise "Modüler-dağıtık Yöntem" olarak adlandırıldı ve yapılan testler sonucunda Modüler-dağıtık Yöntemin "Zombi bilgisayarlar üzerindeki DDoS-öncesi dağıtık hareketleri belirleyen akıllı ve adaptif SBS" olarak çalıştırılabileceğine karar verildi.

Yöntem_2'de konakçılardaki kayıtları periyodik olarak kontrol eden Static Agent'lar merkezdeki MainAgent'a, Ab paketinin "header" bölümünden derlenen kromozomu içeren ikilik dizinin tamamını (IntrusionType) ve "kaynak-IP, hedef-IP, protokol ve paket tipi, zaman bilgileri" gibi gerekli diğer bilgileri içeren sızma-şüphesi mesajı göndermekteydiler. Diğer konakçılardan gelen mesajlardaki intrusionType ve diğer bilgilerin karşılaştırılması sonucunda dağıtık bir saldırı ihtimali ortaya çıkarsa yaratılacak olan Mobile Agent önceden belirlenen iki adet konakçıya gitmekte ve mesaj olarak gönderilmeyen başka bilgileri karşılaştırmaktaydı. Hâlbuki Static Agent'lar tarafından merkeze gönderilen saldırı-şüphesi mesajlarda gerekli olan tüm bilgiler vardır ve MainAgent bir Mobile Agent yaratma kararı verdiğinde aslında gerçekten bir dağıtık saldırı yapıldığı kararını da vermektedir. Bu çalışma mantığıyla Yöntem_2'nin Yöntem_1'den hem etkinlik hem de sistem güvenirliliği açısından çok farkı kalmamaktadır.

Ağda neden olduğu yükü azaltarak bir fark oluşturmak amacıyla mesajların içeriğinde "intrusionType" dizisi yer almaması durumunda, sadece "kaynak-IP ve paket tipi" bilgilerine göre Mobile Agent yaratılmak zorunda kalınır ki, aynı "kaynak-IP ve paket tipi" bilgilerine sahip başka bir Mobile Agent'ın yaratılmasına izin verilmez. Bu durumda, aynı "kaynak-IP ve paket tipi" özelliklerinde farklı bir saldırı yapılıyor ise incelemeye tabi olmadan gözden kaçacaktır. "upload" işleminin yapılıp yapılmadığının incelenmesi Static Agent'lar tarafından yapıldığı için merkeze gönderilen mesajlarda paketlerin "dataload" bölümü yer almamaktadır. MainAgent "dağıtık upload" yapılıp yapılmadığını "upload" ile ilgili gelen mesaj bilgilerine göre karar vermektedir.

Yöntem_3'de Yöntem_2'den farklı olarak, merkeze gönderilen saldırı-şüphesi mesajlarından özellikleri birbiriyle uyuşanların sayısının iki adet olması beklenmez; gönderilen her farklı saldırı-şüphesi mesajına karşılık ağda dolaşarak ilgili saldırı

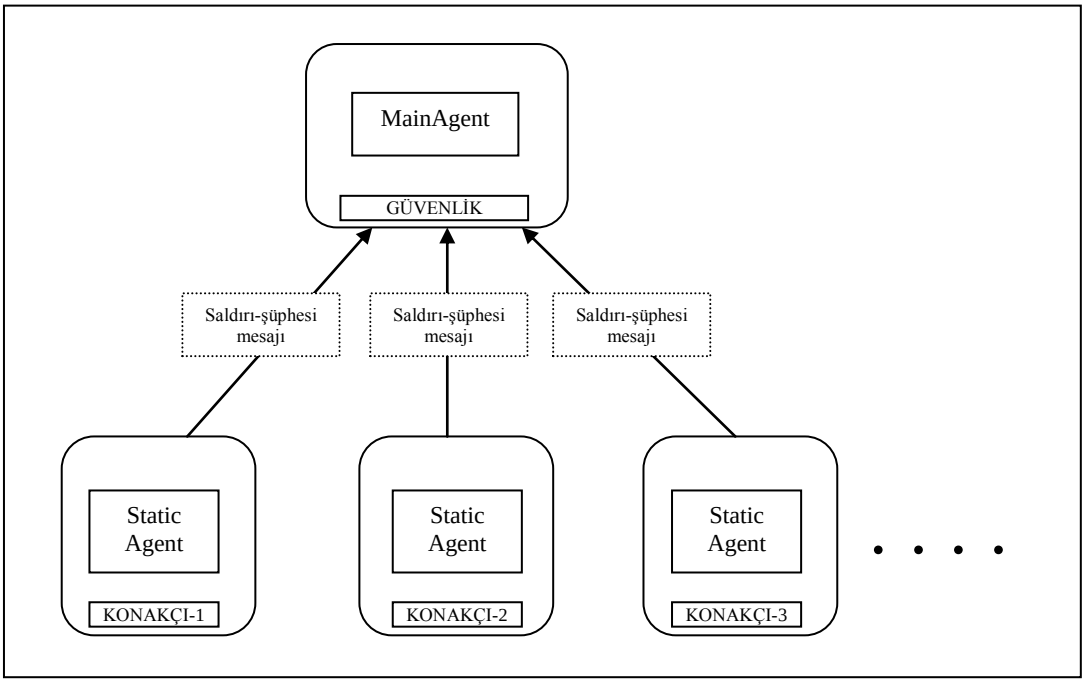
hakkında inceleme yapacak olan Mobile Agent'lar yaratılır. Aynı nitelikte inceleme yapacak olan birden fazla Mobile Agent yaratarak ağ yükünü artırmamak için merkeze gönderilen her saldırı-şüphesi mesaj Yöntem_1 ve Yöntem_2'de olduğu gibi önceki kayıtlı mesajlarla karşılaştırılır; eğer öncekilerden farklı özelliğe sahip ise yeni bir Mobile Agent yaratılır. Önceki mesajların birisiyle aynı özelliklerdeyse yeni Mobile Agent yaratılmaz, ama Yöntem_1'de olduğu gibi, aslında aynı kaynaktan dağıtık bir saldırı yapıldığı anlaşılmış olur. Bu durumda, merkezde belirlenen bu dağıtık saldırıyı belirlemek için yola çıkmış olan Mobile Agent da aynı saldırıyı tespit edecek fakat sistemde gereksiz yere yük oluşturmuş olacaktır. Merkeze gönderilen mesajların boyutunu azaltmak için bazı bilgiler konulmaz ise karşılaştırma yapmak için yeterli bilgi olmayacak ve nerdeyse gelen her mesaj için yeni Mobile Agent yaratılacaktır ki, bu durumda Yöntem_5'den tek farkı Mobile Agent'ların konakçılar tarafından değil de merkezi ünite tarafından yaratılması olacaktır. Dağıtık saldırıya ait ilk saldırı-şüphesi mesajı gönderildikten sonra MainAgent çalışamaz duruma gelmesi durumunda önceden yaratılan Mobile Agent çalışmasına devam edecek ve ilgili dağıtık saldırıyı belirleyecektir. Merkezi üniten olan bu kısmi-bağımsızlık durumu DDoS saldırısını oluşturan dağıtık saldırıların bazıları için etkin olduktan sonra merkezi ünite çalışamaz durumda olduğu için sonraki saldırılar açısından etkin olamayacak ve sistemin güvenirliliği zedelenecektir.

Yöntem_4'de konakçılar şüpheli kayıtlar için bir Mobile Agent yaratmadan önce aynı işi yapacak olan bir Mobile Agent'ın başka bir konakçı tarafından yaratılıp yaratılmadığını öğrenmek için MainAgent'a soru mesajı göndermektedir. Bu soru mesajlarında önceki üç yöntemde olduğu gibi intrusionType dahil diğer bazı bilgiler gönderilmek durumundadır. Yöntem_3 ve Yöntem_5'in birleşimi olan Yöntem_4'de de merkeze gönderilen bilgi yüklü mesajlar nedeniyle vazgeçilen diğer iki yöntem gibi gereksiz yere Mobile Agent yaratma durumu sözkonusudur. Yöntem_3'den vazgeçildiği için Yöntem_4'den de vazgeçilerek benzer yaklaşıma sahip olacak şekilde Yöntem_1 ve Yöntem_5'in birleşimi olan Yöntem_6 kullanılacaktır.

5.1.1 Merkezi Yöntem

MobileAgent ve AlarmAgent kullanılmayan bu yöntemde, konakçılardaki Static Agent'lar merkezdeki MainAgent'a şüpheli olay hakkındaki tüm verilerin yer aldığı

sızma şüphesi mesajları gönderirler. Şekil 5.5'deki modelde gösterildiği gibi, aynı tipte olup da farklı konakçılardan gelen sızma mesajlarının sayısı birden fazla (iki) olursa, güvenlik konsoluna ilgili dağıtık sızma hakkında bilgi verilir.



Şekil 5.5 : Merkezi yöntem modeli.

Sistem ilk başlatıldığında JADE "container" ve "agent" ların yaratıldığı "RmiClient" sınıfı çalışır. Bu sınıftaki "*Runtime.getRuntime().exec("cmd /c start java -javaagent: C:/jade/lib/sizeofag.jar jade.Boot MainAgent: anomaly_realtime_method_1. MainAgent ")* " komutu ile JADE "Main-container " yaratılır ve içerisinde MainAgent çalıştırılır. Daha sonra "*Runtime.getRuntime().exec("cmd /c start java -javaagent:C:/jade/lib/sizeofag.jar jade.Boot -container -host UGUR-PC SA "+i+": anomaly_realtime_method_1.StaticAgent ("+i+")")* " komutu bir döngü içerisinde yirmi kez çalıştırılarak her biri farklı bir Command Prompt'ta çalışan "container" ve Static Agent'lar yaratılır.

Static Agent'ın çalışma mantığı Şekil 5.6'daki algoritmada ifade edilmiştir. "IntrusionController" metodu, iki saniyelik periyotlarla "alarm.ids" dosyasındaki saldırı-şüphesi kayıtlarını incelemekte ve gerekli bilgileri MainAgent'a mesajlamaktadır.

```
class StaticAgent
Begin
 Sistem başlangıç saatini veriseti başlangıç saatine eşitle.

 class IntrusionController extends TickerBehaviour (2 sn)
 Begin
  "LastLineFiles.txt" dosyasından en son hangi saldırı-şüphesi
  kaydının okunduğunu öğren.
  Repeat
   "alarm.ids" dosyasından sıradaki kaydı oku.
   Eğer sistem saati kayıtın gerçekleşme zamanından küçük ise
   döngüden çık.
     if (paketin DATA kısmında UPLOAD işlemi)
       paket tipi= "u"
     Else paket tipi= protokol kodu
   Paketin HEADER bilgilerini içeren mesajı MainAgent'a gönder.
  until ("alarm.ids" dosyası sonu)
   "LastLineFiles.txt" dosyasındaki satır sayısını güncelle.
  End (IntrusionController)
End (StaticAgent)
```

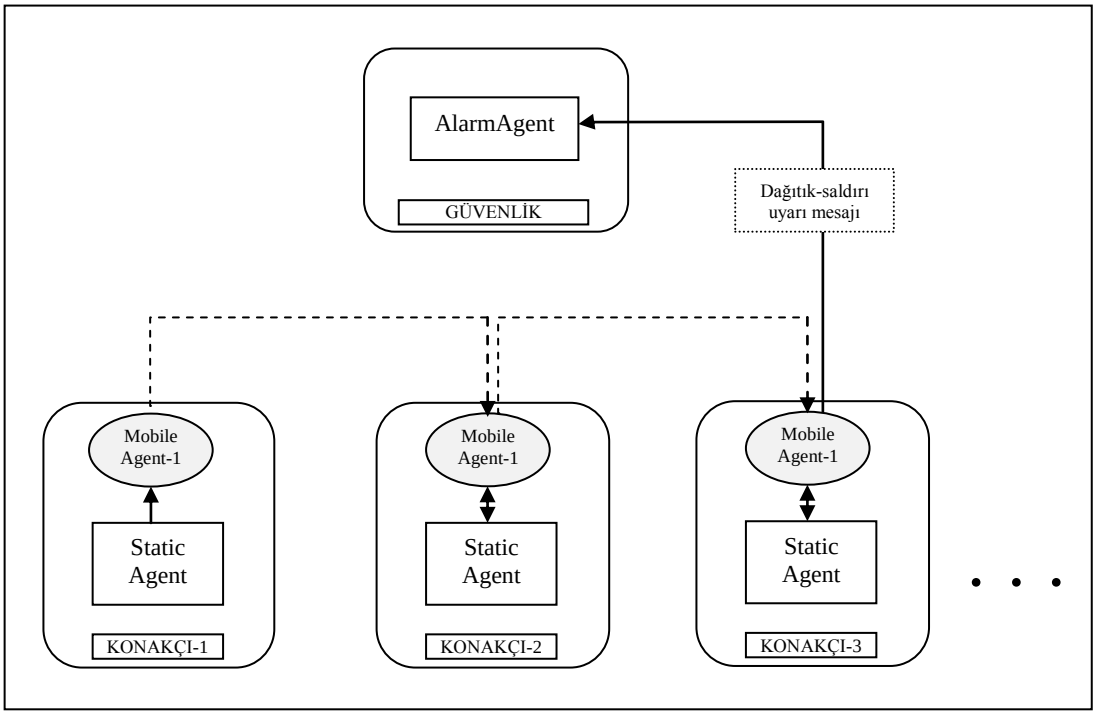
Şekil 5.6 : Merkezi yöntemdeki Static Agent algoritması.

MainAgent, gelen saldırı-şüphesi mesajlarını "HashMap" veri yapısı kullanarak "intrusionType" verilerine göre saklar. Benzer mesajlar aynı "map" içerisinde gruplanır ve güvenlik konsolunda ilgili dağıtık saldırı mesajı yazdırılır. Verisetinin incelenme süresi dolduğunda JADE platformundaki "agent" ve "container"ların tamamını sonlandırma görevi de MainAgent'tadır. MainAgent'ın çalışma mantığı Şekil 5.7'deki algoritmada ifade edilmiştir.

```
class MainAgent
Begin
 class MessageHandler extends CyclicBehaviour
 Begin
   if (verisetinin incelenme süresi doldu)
    JADE platformundaki elemanları sonlandır.
   Else
    İlk mesaj bilgilerini IntrusionTypeMap'e kaydet.
    Repeat
     if (gelen mesaj özellikleri kayıtlı mesajınkilerle
     eşleşiyor)
       if (gelen mesaj kayıtlı mesajdan en fazla 1 saat sonra
       geldi)
         if (gelen mesajı gönderen Static Agent kayıtlı mesajı
         gönderen Static Agent'tan farklıdır)
          Gelen mesajı eşleştiği kayıtlı mesaj grubuna ekle.
          Güvenlik konsolunda dağıtık saldırı bilgisini yazdır.
    until (IntrusionTypeMap sonu)
  End (MessageHandler)
End (MainAgent)
```

Şekil 5.7 : Merkezi yöntemdeki MainAgent algoritması.

5.1.2 Tam-dağıtık Yöntem



Şekil 5.8 : Tam-dağıtık yöntem modeli.

MainAgent'ın hiç kullanılmadığı bu yöntemde, sızma şüphesi belirleyen konakçıdaki Static Agent Şekil 5.8'de görüldüğü gibi ilgili sızmayı araştırmak amacıyla bir Mobile Agent başlatır. Mobile Agent'lar, bütün diğer konakçıları rastgele dolaşarak gerekli sorgulamayı yapar. Aynı özelliklere sahip sızma-şüphesi barındıran bir konakçı daha bulursa taramasını durdurur, ağdaki aynı işi yapan diğer Mobile Agent'ları yayımladığı 'broadcasting' mesajı vasıtasıyla sonlandırır ve AlarmAgent'a bilgi mesajı gönderdikten sonra kendisini sonlandırır. Aksi takdirde, bütün konakçıları sorgulayarak dolaşır ve bir saat sonrasında kendisini sonlandırır.

Sistem ilk başlatıldığında çalışan "RmiClient" sınıfında bu kez Main-container olarak güvenli bir alanda kurulu olduğu varsayılan AlarmAgent yaratılır. Sonrasında, merkezi yöntemde olduğu gibi her biri farklı bir konakçıda yer alan yirmi adet Static Agent, farklı konakçıları modelleyen "command prompt"larda başlatılır. Verisetinin incelenme süresi dolduğunda JADE platformundaki "agent" ve "container"ların tamamını sonlandırma görevi bu yöntem için AlarmAgent'tadır.

Bu yöntemdeki Static Agent'lar merkezi yöntemdekilerden farklıdır, çünkü MainAgent'a gönderilen mesajların yerini Mobile Agent'lar almıştır. Static Agent'in

çalışma mantığı Şekil 5.9'daki algoritmada gösterilmektedir. "Behaviour" sınıfından türeyen sınıfların hepsi birbirlerini engellemeden parallel olarak çalışmaktadırlar.

```
  class StaticAgent
  Begin
   Sistem başlangıç saatini veriseti başlangıç saatine
eşitle.

   class MessageHandler extends CyclicBehaviour
   Begin
     Mobile Agent'lar tarafından gönderilen "broadcasting"
     mesajlarını dinle ve gelen bilgileri kaydet.
   End (MessageHandler)

   class IntrusionController extends TickerBehaviour (2 sn)
   Begin
    "LastLineFiles.txt" dosyasından en son hangi saldırı-
    şüphesi kaydının okunduğunu öğren.
    Repeat
     "alarm.ids" dosyasından sıradaki kaydı oku.
     Eğer sistem saati kayıtın gerçekleşme zamanından küçük
     ise döngüden çık.
       if (benzer işlevde bir Mobile Agent bu Static Agent
       tarafından önceden yaratılmamış)
         if (benzer işlevde bir Mobile Agent başka bir
         Static Agent tarafından daha önce yaratılmamış)
           if (paketin DATA kısmında UPLOAD işlemi)
             paket tipi= "u"
           Else paket tipi= protokol kodu
           Paketin HEADER bilgilerini kaydet.
           Paket ile ilgili Mobile Agent yarat.
    until ("alarm.ids" dosyası sonu)
     "LastLineFiles.txt" dosyasındaki satır sayısını
     güncelle.
    End (IntrusionController)

    class GiveInfo extends CyclicBehaviour
    Begin
      Bu konakçıya gelen Mobile Agent'lar tarafından
      gönderilen "getInfo" mesajlarını dinle.
      Repeat
        "alarm.ids" dosyasından sıradaki kaydı oku.
        if (benzer özelliklerde bir kayıt var)
          Soruyu soran Mobile Agent'a olumlu cevap dön.
          Döngüden çık.
      until ("alarm.ids" dosyası sonu)
    End (GiveInfo)
  End (StaticAgent)
```

Şekil 5.9 : Dağıtık yöntemdeki Static Agent algoritması.

"MessageHandler" adlı "CyclicBehaviour" sınıfı, başka konakçılarda daha önce yaratılan Mobile Agent'lar tarafından gönderilen "broadcasting" mesajlarını dinler ve mesajla gelen bilgileri "HashMap" veri yapısıyla saklar. Mobile Agent

yaratmadan once sakladığı bilgileri kontrol ederek benzer nitelikte bir dağıtık saldırı aramak için ağda dolaşan bir Mobile Agent olduğu halde gereksiz yere kendisi Mobile Agent yaratmaz. "IntrusionController" adlı "TickerBehaviour" sınıfı, "alarm.ids" dosyasındaki saldırı-şüphesi kayıtlarını inceler, önceden bir benzerini kendisi yaratmadıysa ve ağda dolaşan bir benzeri yoksa, ilgili dağıtık saldırıyı aramak üzere bir Mobile Agent yaratır. "GiveInfo" adlı "CyclicBehaviour" sınıfı ise, ağdaki bir Mobile Agent Static Agent'ın bulunduğu "container"a geldikten sonra bilgi almak amacıyla Static Agent'a gönderdiği soru mesajını cevaplar. Mobile Agent tarafından gönderilen saldırı bilgileriyle kendi "alarm.ids" kayıtlarındaki bilgilerle eşleştirir. Eğer bir benzerlik bulursa Mobile Agent'ı haberdar eder.

Mobile Agent yaratıldıktan sonra bir daha yaratıldığı konakçıya uğramayacak şekilde bir saat boyunca ağdaki konakçıları rastgele sırada dolaşmaya başlar. Kendisinin incelediği saldırıya benzer bir saldırıyı incelemek amacıyla başka bir Mobile Agent yaratılmasını engellemek veya kendisinden sonra yaratılmış olan varsa onları durdurmak için sisteme "broadcasting" mesajı gönderir. "MessageHandler" adlı "CyclicBehaviour" sınıfı, başka konakçılarda daha önce yaratılan Mobile Agent'lar tarafından gönderilen "broadcasting" mesajlarını dinler ve kendi özellikleriyle eşleşiyorsa ait olduğu Mobile Agent'ı sonlandırır.

"MyBehaviour" adlı "SequentialBehaviour" sınıfı, sırasıyla "MoveNext" ve "GetInfo" adlı "OneShotBehaviour" sınıflarını yaratır. "MoveNext" sınıfında Mobile Agent'ın gideceği bir sonraki konakçı belirlenir ve Mobile Agent oraya taşınır. "GetInfo" sınıfında ise Mobile Agent uğradığı konakçıdaki Static Agent'a soru mesajı göndererek "alarm.ids" dosyasında kayıtlı benzer bir saldırı-şüphesi olup olmadığını taramasını ister. Static Agent'tan olumlu bir cevap gelmesi durumunda Alarm Agent'a tespit edilen dağıtık saldırı hakkında gerekli bilgileri içeren bir mesaj gönderir ve kendisini sonlandırır. Mobile Agent çalışma mantığı Şekil 5.10'daki algoritmada gösterilmektedir.

Alarm Agent'taki "UpdateAlarms" adlı "CyclicBehaviour" sınıfı, kendisine Mobile Agent'lar tarafından gönderilen dağıtık saldırı mesajlarının bilgilerini derleyerek güvenlik konsolunda görüntülenmesini sağlar. Bir dağıtık saldırı sonrası dağıtık "upload" mesajı gelirse, belirtilen konakçılar üzerinden DDoS saldırısı başlatılacağı kararını verir ve bu durumu güvenlik konsoluna yansıtır.

```
class MobileAgent
Begin
 class MessageHandler extends CyclicBehaviour
 Begin
   Mobile Agent'lar tarafından gönderilen "broadcasting"
   mesajlarını dinle.
   İf (mesaj özellikleri kendi özellikleriyle eşleşiyor)
     Kendini sonlandır.
   EndIf
 End (MessageHandler)

 class myBehaviour extends SequentialBehaviour
 Begin
  class MoveNext extends OneShotBehaviour
  Begin
   Gidilecek olan bir sonraki konakçıyı belirle.
   Belirlenen konakçıya git.
  End (MoveNext)
  class GetInfo extends OneShotBehaviour
  Begin
    Şu an üzerinde olunan konakçıda incelenen saldırı
    benzeri bir saldırı kaydı var mı?
    İf(benzer bir kayıt var)
      AlarmAgent'a "dağıtık saldırı var" mesajı gönder.
      Kendini sonlandır.
    Else ağda 1 saat boyunca dolaşmaya devam et.
  End (GiveInfo)
 End (myBehaviour)
End (MobileAgent)
```

Şekil 5.10 : Dağıtık yöntemdeki Mobile Agent algoritması.

5.1.3 Modüler-dağıtık Yöntem

Bu yöntemde, merkezi ünite çalışır durumdayken düşük ağ yükü avantajı dolayısıyla "merkezi yöntem"in çalıştığı "mod-1" hakimken, merkezi unite çalışamaz olduğunda merkezden bağımsız olduğu için yüksek güvenirlilik avantajı olan "dağıtık yöntem"in çalıştığı "mod-2" hakim olur. Mod-1'de, MainAgent, Static Agent'lardan gönderilen her mesaja karşılık olarak, çalışır durumda olduğu bilgisini içeren bir cevap mesajı gönderir. Eğer ki, bu mesaj 2 saniye içerisinde ilgili Static Agent'a ulaşmaz ise, o konakçı mod-2'ye geçer ve "dağıtık yöntem" ile çalışmasına devam eder. Aynı zamanda, diğer bütün konakçılara mod-2'ye geçmeleri gerektiğine dair bir "broadcasting" mesajı gönderir. Bu uyarı mesajını alan Static Agent, eğer henüz kendiliğinden mod-2'ye geçmemişse, mod-2'ye geçer ve dağıtık olarak çalışmasına devam eder. Bu yöntem başlatıldığında ilk çalışan sınıf olan "RmiClient" sınıfında "Main-container" olarak AlarmAgent yaratılır. Ayrı bir konakçıda MainAgent

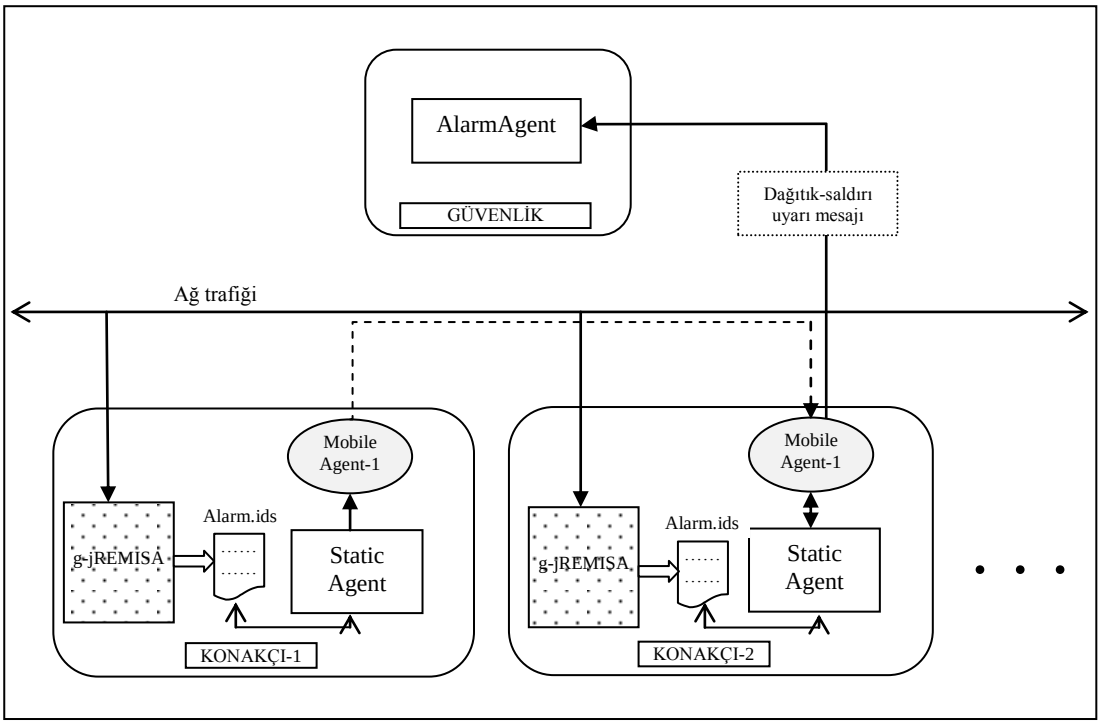
yaratılır ve sonrasında yirmi adet “container” (konakçı) ve Static Agent’lar yaratılır. MainAgent ve Alarm Agent’ın güvenli bir alanda kurulu olduğu varsayılmıştır.

“Merkezi yöntem” ve “dağıtık yöntem” sınıflarının beraber kullanıldığı bu yöntemdeki Static Agent algoritmasında ilk başta “merkezi yöntem”de anlatılan “IntrusionController” adlı “TickerBehavior” sınıfı, MainAgent’a gönderilen her saldırı-şüphesi mesajı sonrasında yaratılan “MessageWaiter” adlı “Behaviour” sınıfı ve diğer konakçılardan gelebilecek mod değiştirme bilgisini dinleyerek kendi konakçısını “mod-2”ye geçiren “ModeControl” adlı “CyclicBehaviour” sınıfı çalışmaktadır. Başka konakçılardan gelen uyarı mesajıyla veya MainAgent’a gönderdiği mesaja zamanında yanıt alamaması nedeniyle kendiliğinden “mod-2”ye geçtiğinde, “dağıtık yöntem”de anlatılan “MessageHandler” ve “GiveInfo” sınıflarıyla, yine aynı yöntemdeki “IntrusionController” sınıfının yerini alan “Mode2Behaviour” adlı “TickerBehavior” sınıfı çalışmaya başlar.

Yöntemin her iki durumda da çalışmasını gözlemleyebilmek amacıyla verisetinin farklı zamanlarında MainAgent çalışamaz duruma getirilmiştir. Bu senaryolar ileriki başlıklarda ele alınmaktadır.

5.2 Sistemin Gerçeklenmesi

“DDoS-öncesi Belirleme Sistemi”, dördüncü bölümde anlatılan “g-jREMISA” adlı anomali-temelli SBS ile bu bölümde anlatılan “modüler-dağıtık yöntem”le çalışan dağıtık yapının birleşiminden oluşmaktadır. Bu sistemin nasıl gerçekleneceği Şekil 5.11’de modellenmiştir. Öncelikle, izlenen her konakçıya bütünleşik sistem bir paket halinde kurulacaktır. Dağıtık olarak izlenecek olan bu konakçılar, daha çok içeriden olan saldırılara açık olan birer yerel ağ düğümü veya bir Internet Hizmet Sağlayıcısı (İHS) tarafından kendilerine internet bağlantısı sunulan ev kullanıcıları olarak düşünülebilir. Her iki durumda da bir DDoS saldırısında aracı olarak kullanılmaya aday bir “botnet” üyesi “zombi” bilgisayar olma ihtimalleri bulunmaktadır. “Modüler-dağıtık yöntem”de merkezi yönetici konumundaki “MainAgent” ve güvenlik konsolunu barındıran “AlarmAgent” birbirinden ve diğer konakçılardan ayrı olarak güvenlikli bir alanda kurulacak ve gelişen durumlara göre belirlenen modda çalışmasına devam edecektir. Şekil 5.11’de “mod-2” çalışması figüre edilmiştir.

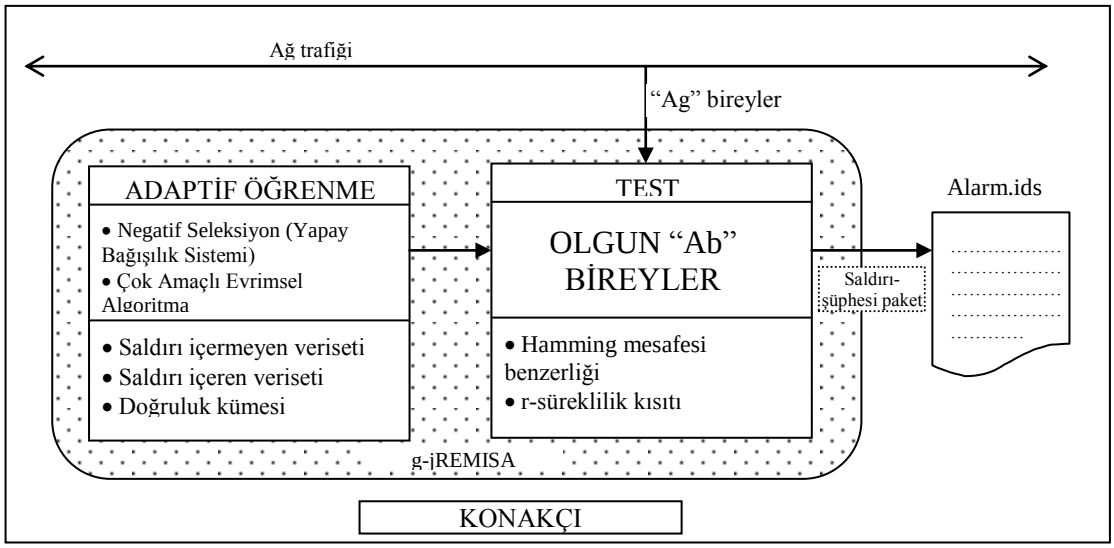


Şekil 5.11 : DDoS-öncesi Belirleme Sistemi'nin gerçeklenmesi.

Konakçıya gelen her ağ paketi öncelikle g-jREMISA modülü tarafından değerlendirilerek anomali nitelikte bir sızma-şüphesi olup olmadığına karar verilecektir. Saldırı-şüphesi paket bilgileri konakçılardaki "alarm.ids" dosyasına kayıt edilecektir. Kısa periyotlarla bu log kayıtlarını kontrol eden Static Agent'lar geçerli olan güvenlik moduna göre kayıtlı paket bilgilerini ya MainAgent'a mesajlayacak veya kendileri o kayıta özel Mobile Agent yaratacak ve diğer konakçılarda dolaşmasını sağlayacaktır. "Mod-1"de MainAgent, gelen mesajları değerlendirdikten sonra tespit ettiği dağıtık saldırıları AlarmAgent'a bildirecektir. "Mod-2" deyse dağıtık saldırı kararını ağda dolaşan Mobile Agent'lar verecek ve AlarmAgent'ı haberdar edecektir. Mobile Agent'ların birbirleriyle ve diğer etmenlerle haberleşirken güvenli bir asıllama uyguladıkları varsayılmaktadır. AlarmAgent'ın görevi de bu dağıtık saldırı bilgilerini güvenlik konsolünde görüntüleyerek güvenlik yöneticisini uyarmaktır.

G-jREMISA modülünün iç yapısı Şekil 5.12'de gösterildiği gibi iki ana bölümden oluşmaktadır. "Öğrenme" işleminin yapıdığı ilk bölüm, bütünleşik sistem konakçılara kurulmadan önce çevrim-dışı olarak çalıştırılmış ve konakçıya gelen her ağ paketini inceleyen "Test" bölümünde kullanılmak üzere "olgun Ab bireyler" oluşturmuştur. Farklı ağ verisetlerine başarılı cevaplar verdiği önceki bölümlerde kanıtlanan g-jREMISA, konakçıya kurulduğunda ilk olarak hazır bulduğu "olgun Ab

bireyler" ile test yapmaya başlayacaktır. İlerleyen zamanlarda "adaptif öğrenme" modülünü çalıştırıp güvenlik yöneticisinden ve kullanıcılardan aldığı geri-dönüşümleri kullanarak daha güncel "olgun Ab bireyler"le yoluna devam edecektir.



Şekil 5.12 : G-jREMISA modülünün iç yapısı.

G-jREMISA'nın gelen bir paketi inceleme süresi ortalama 2 milisaniye olarak ölçülmüştür. Ağ analiz aracı kullanılarak internetten bir video indirirken dahi paketlerin minimum 5 milisaniye aralıklı olarak (~200 paket/sn) geldiği hesaplanmıştır. Adaptif öğrenme modülü trafiğin akışını engellemeden paralel olarak çalışabilmektedir. Bu durumda g-jREMISA'nın gelen trafiği yavaşlatmayacağı söylenebilir.

5.3 Adaptif Öğrenme

Adaptif öğrenme yaklaşımında, bütünleşik sistemdeki g-jREMISA modülünün gelen paketler hakkında yaptığı değerlendirmelerin büyük çoğunluğu doğru olmakla beraber, sistemin yeni tanımlanan saldırı türlerine karşı uyarlanabilme ve yanlışlıkla anomali olarak değerlendirilen trafiğin (yanlış pozitif) "normal profile" eklenebilmesi özellikleri vardır.

G-jREMISA, gelen paketleri incelemeden once "doğruluk kümesi" dosyasının güncellenip güncellenmediğini kontrol eder. Doğruluk kümesine olan eklemeler, yeni saldırı tanımları güncellemesi yapan güvenlik yöneticisi tarafından veya sistemin bağlı olduğu bir şebeke tarafından yapılacaktır. Doğruluk kümesiyle beraber "saldırı içeren veriseti" de güncellenecektir. Eğer bir güncelleme fark edilirse

“updateAb” modülü çalıştırılacaktır. Bu modülde, g-jREMISA’nın ÇAEA aşaması yeni “doğruluk kümesi” ve “saldırı içeren veriseti” ile mevcut “olgun Ab bireyler” kullanılarak çalıştırılacak ve yeni “olgun Ab bireyler” yaratılacaktır. Bundan sonraki değerlendirmelerde yeni saldırı türleri çok daha doğru sonuçlarla tanınabilecek, eğer varsa önceden yanlışlıkla sistemin “kendinden-olan” olarak kabul edip “yanlış negatif” değerlendirme yaptığı saldırılar artık “kendinden-olmayan” olarak değerlendirilecektir.

“updateAb” modülünün bir diğer fonksiyonu da, “yanlış pozitif” uyarılar verilmesine neden olan değerlendirmelerin güvenlik yöneticisi veya kullanıcı tarafından belirlendikten sonra “normal profili” oluşturan “saldırı içermeyen veriseti”nin güncellenmesine izin vermesidir. Bu verisetinin güncellendiğini fark eden g-jREMISA “updateAb” modülü vasıtasıyla once “Negatif Seleksiyon” sonra da ÇAEA aşamalarını yeni “saldırı içermeyen veriseti”ne gore tekrar çalıştıracak ve daha güncel “olgun Ab bireyler” ile yoluna devam edecektir. Böylece “yanlış pozitif” değerlendirmeler yerini “doğru negatif” değerlendirmelere bırakacaktır.

Dördüncü bölümde detayları anlatılan g-jREMISA algoritmasındaki “test” aşaması, “olgun Ab bireyler”in “saldırı içeren verisetleri”ne verdikleri cevapların kullanılan verisetine ait olan “doğruluk kümesi” ile karşılaştırılması sonucunda oluşan TP/FP değerlerini ortaya koymaktadır. Gerçek sistemde yapılan testlerde “doğruluk kümesi” kullanılamadığı için TP/FP sonuçlar ortaya konamayacak, ancak kullanıcı ve güvenlik yöneticilerin yapacağı değerlendirme sonunda g-jREMISA tarafından verilen kararın doğru olup olmadığı değerlendirilebilecektir.

“Doğruluk kümesi”ne ve “saldırı içeren verisetleri”ne yapılacak eklemeler mevcut kayıtların sonuna yapılacak ve “test” aşamasında henüz kontrol edilmemiş ve zaten “yanlış negatif” değerlendirmeye neden olmamış olacaktır. Sırası geldiğinde güncellenen “olgun Ab bireyler”i tarafından incelenecek ve “doğru pozitif” sonuç verdirecektir. Bu durumda adaptif öğrenmenin etkisi sonuçlara yansımayacaktır. Adaptif öğrenmeyi test etmenin bir yolu, aynı “saldırı içeren veriseti” ile önce bazı saldırı paketlerinin yer almadığı “eksik doğruluk kümesi” ile test yapmak ve sonra aynı testleri “tam doğruluk kümesi” ile yaparak değerlendirme sonuçlarını karşılaştırmaktır. Çizelge 5.1’de MIT DARPA 1999 veriseti dördüncü hafta Pazartesi günü “saldırı içeren veriseti” ile eğitilen g-jREMISA’nın aynı haftanın Çarşamba günüki “saldırı içeren veriseti”ne verdiği cevapların adaptif öğrenme öncesi ve

sonrasındaki TP/FP değerlendirmeleri yer almaktadır. Adaptif öğrenme öncesi yapılan testlerde ÇAEA aşamasında "eksik doğruluk kümesi", adaptif öğrenme sonrası yapılan testlerde ise ÇAEA aşamasında "tam doğruluk kümesi" aynı "saldırı içeren veriseti" ile beraber kullanılmıştır. "Doğru pozitif" oranlarındaki yaklaşık % 3'lük iyileşme göze çarpmaktadır.

Çizelge 5.1 : Adaptif öğrenme test sonuçları.

Eğitim		**Test (hafta-4)**	**Doğru Pozitif (%)**	**Yanlış Pozitif (%)**	
Negatif Seleksiyon (hafta-1)	**ÇAEA (hafta-4)**				
Beş günün bileşimi	Pazartesi	Çarşamba	94,72	0,0025	**Adaptif öğrenme öncesi**
			97,60	0,0038	**Adaptif öğrenme sonrası**

5.4 Uygulama ve Deneyler

"DDoS-öncesi Belirleme Sistemi"nin başarısını ve işlevselliğini görebilmek için literatürdeki tek DDoS senaryolu verisetler olan MIT DARPA 2000 LLS_DDOS 1.0 ve 2.0.2 verisetleri ile ayrı ayrı testler yapılmış ve sonuçlar değerlendirilmiştir. Bütünleşik sistemdeki "modüler-dağıtık" yöntemin ve onu oluşturan iki alt yöntemin bu verisetlerindeki DDoS öncesi dağıtık saldırıları belirleme süreleri ve bu sürede ağda neden oldukları yük değerleri ortaya konulmuş ve yöntemler güvenirlilik açısından tekrar ele alınmıştır. Sistemin ilk ayağı olan g-jREMISA'nın LLS_DDOS verisetlerindeki saldırıları doğru tespit edebilme oranları farklı senaryolar kapsamında değerlendirilerek g-jREMISA'nın kendi ortamındaki ve eğitildiği verisetleri dışındaki ortamlarda gösterdiği performans gözlemlenmiştir.

5.4.1 LLS_DDOS Verisetleri

MIT DARPA 2000 LLS_DDOS verisetlerinin hazırlanma ortamları ve önceki testlerde kullanılmış olan 1.0 versiyonu hakkında detaylı bilgi 3.2.3 numaralı bölümde verilmiş olup, bu bölümde ilk defa kullanılacak olan 2.0.2 versiyonu daha detaylı olarak ele alınacaktır. Verisetinin 1.0 versiyonundaki saldırıcı, simule ağdaki üç Solaris konakçıya yönetici erişimi sağlamak için Solaris sadmind açığını, saldırıyı başlatmak için de Mstream DDoS aracını kullanmaktadır. Üç ara konakçıya da birer Mstream "server" kurulurken, bunlardan bir tanesine "server"ları kontrol eden bir Mstream "master" kurulmuştur. DDoS sadırısı bu "server"lar tarafından eşzamanlı

olarak başlatılmıştır. Saldırı senaryosunun beş aşaması olup bu aşamaların gerçekleşme zamanları Çizelge 3.1'de ifade edilmiştir.

Verisetinin 2.0.2 ve 1.0 versiyonları arasındaki temel fark, 2.0.2 versiyonda saldırıcının (202.077.162.213) konakçı, platform ve işletim sistemi yoklamalarını IP adreslerini ve rpc (remote procedure call) portlarını tarayarak değil de DNS HINFO (Host INFO) sorgulamalarıyla gerçekleştirmesi ve her bir konakçıya ayrı ayrı saldırmak yerine önce Eyrie Hava Üssündeki konakçılardan bir tanesine (DNS Sunucu) girmesi ve sonra oradan diğerlerine nüfuz etmeye çalışmasıdır [78].

16 Nisan 2000 günü 14:45 ile 16:27 arasında yaklaşık bir buçuk saat boyunca oluşturulmuş olan saldırı senaryosunun beş aşaması vardır:

1. Eyrie Üssünün DNS sunucusu olan mill.eyrie.af.mil (172.16.115.20) konakçısının HINFO sorgularıyla yoklanması.

2. Sadmind açığı kullanılarak mill.eyrie.af.mil konakçısına nüfuz edilmesi.

3. Daha fazla Eyrie konakçısına girebilmek için, mstream DDoS yazılımının ve saldırı betiklerinin FTP Upload ile yüklenmesi.

4. Diğer Eyrie konakçılarına saldırı başlatılması: mill.eyrie.af.mil konakçısına Telnet ile bağlanılması, "DDoS master" olarak kurulması ve diğer Eyrie konakçılarına yoklama ve saldırıların başlatılması. İki adet nüfuz etme denemesi yapılmıştır: robin.eyrie.af.mil (172.16.112.207) (HINFO sorgulamasında Solaris olarak bildirilen bir Linux konakçı, işlem başarısız olmuştur!) ve pascal.eyrie.af.mil (172.16.112.50) (konakçı Solaris olduğu için işlem başarılı olmuştur!).

5. DDoS saldırısının başlatılması: mill konakçısına ve localhost 6723 numaralı port'a Telnet ile bağlanılması, "master"a bağlanılması ve "mstream 131.84.1.31 5" komutuyla "www.af.mil" sitesine 5 saniyelik saldırının başlatılması.

Çizelge 5.2'de saldırı aşamalarının gerçekleşme zamanları (Türkiye saat diliminde) detaylı olarak yer almaktadır. 4a numaralı dağıtık saldırı sonrasında 3 numaralı saldırı ile başlayan ve 4c numaralı saldırıyla sonlanan "dağıtık UPLOAD" işlemi belirlenerek DDoS saldırısı yapılacağı kararı verilmekte ve güvenlik yöneticisine bilgi verilmektedir. "Dağıtık UPLOAD" işlemi, ilk olarak saldırıcı tarafından DNS sunucuya yapılmakta, daha sonra ise DNS sunucu tarafından pascal adlı konakçıya

yapılmaktadır. Bu saldırı aşamaları arasında kurulan bağlantı vasıtasıyla bir karar verilmekte olduğu için, DDoS uyarısı verdirmeden once geçecek olan minimum sure 15 dakika 5.62 saniye olarak ortaya çıkmaktadır.

Çizelge 5.2 :LLS_DDOS 2.0.2 saldırıların gerçekleşme zamanları.

Saldırı Aşamaları	Saldırı tipleri	Saldırı yapılan konakçı sayısı	Saldırının ilk konakçıya yapıldığı zaman	Saldırının ikinci konakçıya yapıldığı zaman
1	DNS Sunucunun yoklanması	1	22:05:44.67	
2	DNS Sunucuya nüfuz edilmesi	1	22:17:44.34	
3	DNS Sunucuya FTP Upload ile Mstream'in yüklenmesi	1	22:29:48.13	
4a	Diğer konakçıların DNS Sunucu tarafından yoklanması	2	22:42:51.45	22:42:51.59
4b	DNS Sunucu tarafından diğer konakçılara nüfuz edilmesi	2	22:44:53.25	22:44:53.57
4c	DNS Sunucu tarafından diğer konakçılara FTP Upload ile Mstream'in yüklenmesi	1	22:44:53.75	
MİNİMUM SALDIRI TESPİT SÜRESİ = 15 DK 5.62 SN = 905,62 SN				
5	DDoS saldırısının başlatılması	2	23:05:36.26	23:05:36.57

5.4.2 G-jREMISA Testleri

5.4.2.1 Test tasarımları

Testler, Windows XP SP3 ile çalışan Pentium Core 2 Duo 2.4 GHz bilgisayarlar ile gerçekleştirilmiştir. MIT DARPA 2000 LLS_DDOS verisetinin 1.0 ve 2.0.2 versiyonları ile 1999 verisetinin farklı günlerindeki veriler, değişik kompozisyonlarla denenmiş ve g-jREMISA algoritmasının performansı değerlendirilmiştir.

Testlerde önceden kullanılan MIT DARPA 2000 LLS_DDOS verisetinin 1.0 versiyonunda olduğu gibi, ilk defa kullanılacak olan 2.0.2 versiyonuna ait "doğruluk kümesi" ve "saldırı içermeyen veriseti" ilgili web sitesinde yer almamaktadır. Saldırı

senaryosu hakkında web sitesinde verilen bilgilerden faydalınarak, Ethereal [72] paket analiz programı ve g-jREMISA'nın "Packet-options" bölümü vasıtasıyla eksik olan bu veriler elde edilmiştir. Öncelikle, verisetinde TCP/UDP/ICMP dışında kalan paketler Ethereal'da yapılan bir süzme işlemiyle elendi. Verisetinin web sayfasında yer alan saldırı bilgileri ve xml kayıtlarından alınan kaynak-IP ve hedef-IP bilgileri g-jREMISA'nın "Packet-options" bölümündeki "Filter" menüsüne girilerek "doğruluk kümesi"ni oluşturacak olan taslak xml dosyaları oluşturuldu. Taslak dosyada yer alan paket numaraları, Ethereal programındaki paket numaralarıyla eşleştirilerek xml dosyalarına son halleri verildi. En sonunda, parça parça olan bu dosyalar birleştirilerek "doğruluk kümesi" dosyası oluşturuldu. "Doğruluk kümesi"nde yer alan paketler Ethereal'da yapılan süzme işlemiyle "saldırı içeren veriseti"nden çıkartıldı ve böylece "saldırı içermeyen veriseti" oluşturuldu.

5.4.2.2 Sonuçlar

Hem adaptif öğrenme (negative seleksiyon ve ÇAEA) hem de test aşamalarında MIT DARPA 2000 LLS_DDOS 1.0 veriseti kullanılarak yapılan testler 4.2.1 numaralı bölümde açıklanmış olup; TCP, UDP ve ICMP popülasyon büyüklüklerinin sırasıyla 300, 100 ve 100, r-sürekli değerin 10 ve benzerlik eşiği değerinin %38 - %44 aralığında olduğu şartlarda ortalama olarak %100 doğru pozitif (TP) ve %0 yanlış pozitif (FP) oranları elde elde edilmiştir.

Çizelge 5.3 : Farklı verisetli testlerin sonuçları.

Adaptif Öğrenme		**Test (hafta-4)**	**Doğru Pozitif (%)**	**Yanlış Pozitif (%)**
Negatif Seleksiyon (hafta-1)	**ÇAEA (hafta-4)**			
LLSDDOS 1.0	LLSDDOS 1.0	LLSDDOS 2.0.2	100	0,07
LLSDDOS 2.0.2	LLSDDOS 2.0.2	LLSDDOS 1.0	100	0,005
LLSDDOS 2.0.2	LLSDDOS 2.0.2	1999-4 Pazartesi	99,47	0,28
LLSDDOS 2.0.2	LLSDDOS 2.0.2	1999-4 Çarşamba	100	0,02
LLSDDOS 2.0.2	LLSDDOS 2.0.2	1999-4 Perşembe	100	0,06
LLSDDOS 2.0.2	LLSDDOS 2.0.2	1999-4 Cuma	100	0,009
1999-4 Günlerin Bileşimi	1999-4 Pazartesi	LLSDDOS 2.0.2	99,61	0,12
1999-4 Günlerin Bileşimi	1999-4 Çarşamba	LLSDDOS 2.0.2	99,22	0,09

TCP, UDP ve ICMP popülasyon büyüklüklerinin sırasıyla 300, 100 ve 100, r-sürekli değerinin 15 ve benzerlik eşiği değerinin %50 alınarak hem adaptif öğrenme hem de test aşamalarında MIT DARPA 2000 LLS_DDOS 2.0.2 verisetinin kullanıldığı testlerde de yirmi koşturma sonrasında ortalama olarak %100 doğru pozitif (TP) ve %0 yanlış pozitif (FP) oranları elde elde edilmiştir.

Dördüncü bölümdeki testlerde olduğu gibi, sistemin daha gerçekçi durumlardaki performansını ölçmek için adaptif öğrenme ve test aşamalarında birbirinden farklı verisetleri kullanılarak da testler yapılmıştır. Çizelge 5.3'de gösterildiği gibi bu tür testlerde de %100'e yakın TP ve yaklaşık %0 FP sonuçlar elde edilmiştir.

5.4.3 DDoS-öncesi Dağıtık Hareketleri Belirleme Sistemi Testleri

Altı farklı yöntem ile gerçeklenen imza-temelli dağıtık SBS'in SNORT tarafından oluşturulan "alarm.ids" kayıtlarını kullanarak MIT DARPA 2000 LLS_DDOS 1.0 verisetindeki DDoS öncesi dağıtık hareketleri dördüncü aşamadan önce belirleyerek güvenlik konsolunda görüntülediği uygulama, tezin üçüncü bölümünde detaylı olarak anlatılmıştır. Yapılan testlerde yöntemlerin her birinin DDoS uyarısı verdirme süreleri, ağda neden oldukları yük ve merkezi üniteden bağımsızlık olarak değerlendirilen güvenirlilik seviyeleri ölçülmüş ve Yöntem_4 en optimum sonuçlara sahip olan yöntem olarak seçilmiştir.

Bu bölümde, imza-temelli dağıtık SBS'in yerini anomali-temelli dağıtık SBS; SNORT'un yerini g-jREMISA almış olup, DDoS öncesi dağıtık hareketlerden olan "dağıtık UPLOAD" işlemi sonrasında güvenlik yöneticisine DDoS uyarısı verdirilmektedir. SNORT tarafından imza-temelli yöntemle tespit edilemeyen bu saldırı aşaması, anomali-temelli yöntemle tespit edilebilmektedir. Bir DDoS saldırısı başlatılmadan önce mutlaka bir "dağıtık UPLOAD" yapılması gerekliliğinden hareketle DDoS uyarısı verdirmeden önce bu aşamanın yapılması beklenmektedir.

Geliştirilen “modüler-dağıtık yöntem” ve onu oluşturan “merkezi yöntem” ile “tam-dağıtık yöntem”, MIT DARPA 2000 LLS_DDOS 1.0 ve 2.0.2 verisetleriyle ayrı ayrı çalışan g-jREMISA’nın ürettiği “alarm.ids” kayıtlarını kullanarak test edilmiş, DDoS uyarısı verdirme süreleri ve o süre içerisinde ağda neden oldukları yük değerleri ölçülmüştür. Testler, Windows XP SP3 işletim sistemi ile çalışan Pentium Core 2 Duo 2.4 GHz bilgisayarlarda gerçekleştirildi. Güvenirlik ve hız açısından en optimum çözüm olarak ortaya konulan “modüler-dağıtık yöntem”in ağda neden olduğu yük ayrıca değerlendirilmiştir.

5.4.3.1 Test tasarımları

LLS_DDOS 1.0

Bu verisetinde ilk dağıtık saldırı olan “ICMP PING” işleminin ilk konakçıya yapılması ile “dağıtık UPLOAD” işleminin ikinci konakçıya yapılması arasında geçen süre 59 dakika 45 saniyedir. DDoS öncesi dağıtık hareketlerin aşamaları Çizelge 5.4’de SNORT tarafından verilen isimler kullanılarak gösterilmiştir. Veriseti, yaklaşık 3 saatlik bir süreyi kapsamasına rağmen DDoS saldırısında ilgilenilen aşamalar 1 saatten daha az bir sürede gerçekleştiği için, testler, bu aşamaların olduğu yaklaşık bir saatlik zaman dilimi boyunca koşturulmuştur.

Çizelge 5.4 : LLS_DDOS 1.0 saldırılarının aşamaları.

Saldırı Aşamaları	Saldırı adları	Saldırı yapılan konakçı sayısı	Saldırının ilk konakçıya yapıldığı zaman	Saldırının ikinci konakçıya yapıldığı zaman	Saldırının son konakçıya yapıldığı zaman
1	ICMP PING	20	16:51:36.14	16:51:36.52	16:52:00.82
2a	RPC portmap sadmind request UDP	11	17:08:07.35	17:08:07.50	17:34:36.43
2b	RPC sadmind UDP Ping	3	17:08:07.36	17:15:10.03	17:15:10.10
3a	RPC sadmind query with root credentials attempt	3	17:33:10.62	17:34:36.45	17:34:49.00
3b	RPC sadmind UDP NETMGT_PROC SERVICE CLIENT_DOMAIN overflow attack	3	17:33:10.62	17:34:36.45	17:34:49.00
4	RSERVICES rsh root (UPLOAD)	3	17:50:02.08	17:50:21.30	17:50:38.25
MİNİMUM SALDIRI TESPİT SÜRESİ = 59 DAKİKA 45.16 SANİYE = 3525,16 SN					

"Modüler-dağıtık yöntem"de MainAgent çalışır durumdayken mod-1, çalışamaz durumdayken mod-2'nin etkin olacağı önceki bölümlerde anlatılmıştı. Bu durumu test edebilmek için yönteme ait program çalışmasını sürdürürken farklı zamanlarda MainAgent sonlandırılarak sistemin otomatikman mod-2'ye geçmesi sağlanmıştır. MainAgent'ın sonlandırma zamanları şu şekildedir:

- 1'inci aşamadan önce (16:50),
- 2'inci aşamadan önce (17:01),
- 3'üncü aşamadan önce (17:28),
- 4'üncü aşamadan önce (17:41),
- 4'üncü aşamadan sonra (18:11).

LLS_DDOS 2.0.2

Çizelge 5.2'de gösterildiği gibi, bu verisetinde DDoS uyarısı verdirmeden önce geçecek olan minimum sure 15 dakika 5.62 saniye olup, bu süre ilk olarak saldırıcı tarafından DNS sunucuya yapılan daha sonra ise DNS sunucu tarafından pascal adlı konakçıya yapılan "dağıtık UPLOAD" işlemi tarafından belirlenmektedir. Veriseti, 1,5 saati biraz aşkın bir süreyi kapsamasına rağmen DDoS saldırı aşamaları 1 saatten daha az bir sürede gerçekleştiği için, testler, bu aşamaların olduğu yaklaşık bir saatlik zaman dilimi boyunca koşturulmuştur.

LLS_DDOS 1.0'da olduğu gibi, "Modüler-dağıtık yöntem"de MainAgent çalışamaz durumdayken etkin olan mod-2'nin çalışmasını test edebilmek için farklı zamanlarda MainAgent sonlandırılmıştır. MainAgent'ın sonlandırma zamanları şu şekildedir:

- Başlangıçta (22:05),
- 1'inci çeyrekten sonra (22:20),
- 2'inci çeyrekten sonra (22:35),
- 3'üncü çeyrekten sonra (22:50),
- En sonda (23:05).

5.4.3.2 Sonuçlar

LLS_DDOS 1.0

Her bir yöntemin 20'şer kere koşturulması sonucunda Çizelge 5.5'deki sonuçlar elde edilmiştir. Yukarıda belirtildiği gibi, minimum saldırı tespit süresinin 59 dakika 45.16 saniye (3525,16 sn) olduğu değerlendirildiğinde, bütün yöntemlerin ortalama saldırı belirleme sürelerinin, minimum süre (+2,56 - 1,55 sn) içerisinde olduğu görülmektedir.

Çizelge 5.5'de aynı zamanda, test süresince konakçılar arasında gönderilen mesajlar ve ağda dolaşan Mobile Agent'lar tarafından oluşturulan ağ yükü ölçümleri de yer almaktadır. "Merkezi yöntem", Mobile Agent kullanılmadığı için sadece Static Agent'lardan MainAgent'a gönderilen mesajlardan dolayı 0.057 MB'lık yüke neden olurken, "tam-dağıtık yöntem" merkezi ünite olmadığı için sadece Mobile Agent'lar ve AlarmAgent arasındaki mesajlar ve ağda dolaşan Mobile Agent'lardan dolayı 143,96 MB'lık yüke neden olmaktadır. Mobile Agent'lar bir konakçıdan diğerine taşınırken ağ yoluna çıkmakta ve ağda yüke neden olmaktadır. Bundan dolayı ölçümlerde bu husus dikkate alınmıştır. "Modüler-dağıtık yöntem"de diğer iki yöntemde kullanılan mesajlarla beraber yönteme özgün iki adet daha mesaj tipi eklenmiştir. Birisi MainAgent'ın gelen her saldırı-şüphesi mesajına karşılık olarak çalışır durumda olduğunu bildiren "Mes6_AliveMesOfMainA" mesajı, diğeri de mod-2'ye geçiş yapan Static Agent tarafından diğer konakçıları haberdar etmek için gönderilen "Mes6_ModeMesOfSA" mesajıdır. Önceki bölümde anlatılan senaryo dahilinde MainAgent sonlandırılmış olduğu için, bu yöntemdeki Mobile Agent tarafından neden olunan ağ yükü daha azdır. Ortalama olarak 87,5 MB ağ yüküne neden olan Modüler yöntem en iyi durumda "merkezi yöntem"e, en kötü durumda da "tam-dağıtık yöntem"e yakın yük değerleri verecektir.

Bir Mobile Agent'ın ortalama boyutu 100 KB, iki konakçı arasında dolaşma süresi 100 msn olarak ölçülmüştür. Verisetindeki senaryo gereği 20 farklı konakçı ve bir güvenlik konsolü yaratılmış olup, 1000 kadar konakçı barındıran geniş bir ağ kullanılması durumunda dahi; ağda bir saat boyunca dolaşıp kendini sonlandıracak olan MA için durum değişmeyecek, toplamda aynı sayıda konakçıyı dolaşacaktır. En kötü durumda, MA dağıtık saldırı yapılan birinci konakçıyı birinci, ikinci konakçıyı sonuncu sırada dolaşacaktır. Bu durumda 20 konakçıda gecikme 2 sn, 1000

konakçıda 100 sn olacaktır. Fazladan dolaşacağı 1000 konakçı dolayısıyla toplamda 100 MB fazla yük oluşturacaktır, ama oran aynı olacaktır (100MB/1000=2MB/20). "Broadcasting" mesajların sayısı 1000 olacak, fakat yük oranı yine aynı olacaktır.

Çizelge 5.5 : LLS_DDOS 1.0 anomali-temelli test sonuçları.

	Merkezi Yöntem	**Tam-dağıtık Yöntem**	**Modüler-dağıtık Yöntem**
Mes1_SAtoMainA	474760		
Mes5_MAtoALL		246564	
Mes5_MAtoAA		11369	
Mes6_SAtoMainA			165610
Mes6_AliveMesOfMainA			187059
Mes6_MAtoALL			135646
Mes6_toAA			19298
Mes6_ModeMesOfSA			11056
MA_5		1207360795	
MA_6			733496938
TOPLAM(bit)	474760	1207618728	734015607
TOPLAM(KB)	57,95	147414,4	89601,52
TOPLAM(MB)	0.057	143,96	87,5
ORTALAMA SÜRE(sn)	3523,606	3527,72	3523,61

LLS_DDOS 2.0.2

Yirmi koşturma sonrasında elde edilen sonuçlar, Çizelge 5.6'da gösterilmiştir. Yöntemlerin ortalama saldırı belirleme süreleri, minimum süre olan 15 dakika 5.62 saniyenin (+1,32 sn) sınırları içerisinde bulunmaktadır. Çizelge 5.6'da aynı zamanda, test süresince oluşan ağ yükü ölçümleri de yer almaktadır. "Merkezi yöntem" 0.069 MB'lık yüke neden olurken, "tam-dağıtık yöntem" ağda dolaşan Mobile Agent'lardan dolayı 3977,8 MB'lık yüke neden olmaktadır. "Modüler-dağıtık yöntem"de bir senaryo dahilinde MainAgent sonlandırılmış olduğu için ortalama olarak 139,99 MB ağ yükü ortaya çıkmıştır.

Çizelge 5.6 : LLS_DDOS 2.0.2 anomali-temelli test sonuçları.

	Merkezi Yöntem	**Tam-dağıtık Yöntem**	**Modüler-dağıtık Yöntem**
Mes1_SAtoMainA	575205		
Mes5_MAtoALL		253664	
Mes5_MAtoAA		5736	
Mes6_SAtoMainA			76158
Mes6_AliveMesOfMainA			77024
Mes6_MAtoALL			38936
Mes6_toAA			5134
Mes6_ModeMesOfSA			11152
MA_5		33367810742	
MA_6			1174139032
TOPLAM(bit)	575205	33368070142	1174347436
TOPLAM(KB)	70,22	4073250,8	143352,96
TOPLAM(MB)	0.069	3977,8	139,99
ORTALAMA SÜRE(sn)	905,87	906,94	906,76

6. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında, gezgin etmenler ve doğadan esinlenen algoritmalar kullanarak DDoS saldırısı öncesi hareketleri orta katmanda dağıtık olarak belirlemek ve saldırı başarıya ulaşmadan önce güvenlik yöneticilerini haberdar etmek amaçlanmıştır. Bu kapsamda, gezgin etmenler yardımıyla dağıtık yapının kurulması ve doğadan esinlenen algoritmalar yardımıyla anomali-temelli bir SBS oluşturulması ayrı ayrı gerçeklenmiş ve en sonunda, bu iki sistem birleştirilerek yeni saldırıları düşük yanlış pozitif oranlarıyla belirleyebilen, adaptif olarak SBS'i güncelleme imkanına sahip, merkezi üniteden bağımsız çalışabildiği için yüksek güvenirlik sahibi, sızma belirleme işlemini gezgin etmenler vasıtasıyla yaptığı için ağ yükünü hafifleten ve özellikle DDoS saldırılarını kötüye kullanıldıklarından habersiz olan ve "botnet" olarak nitelendirilen orta katmanda dağıtık olarak başarıyla belirleyebilen bir SBS geliştirilmiştir.

Tezin ilk aşamasında, MIT DARPA 2000 LLS_DDOS 1.0 verisetindeki DDoS saldırılarını tespit edebilmek için altı farklı dağıtık sızma belirleme yöntemi tasarlandı ve simule-gerçek zamanlı test ortamında saldırıların gerçekleşme zaman bilgilerini dikkate alarak test edildi. Bu imza-temelli SBS yöntemlerinin birincisi dışında hepsinde gezgin etmenler kullanıldı. Yöntem_1'de Mobile Agent'lar kullanılmaksızın merkezi bir dSBS yer almakta olup bir saat içerisinde farklı konakçılardan aynı kaynaklı, aynı tipte bir sızma-şüphe mesajı alınırsa dağıtık bir sızma yapıldığı kararına varılmakta ve güvenlik yöneticisi haberdar edilmektedir. Yöntem_2'de dağıtık sızma olup olmadığı kararını, yukarıda belirtilen şartlar oluştuğunda MainAgent tarafından yaratılan ve sadece listesindeki iki adet konakçıya giderek ilgili verileri yerinde inceleyen Mobile Agent'lar vermektedirler.

Yöntem_3'de, MainAgent, Mobile Agent yaratmak için gerekli şartları sağlayan mesaj sayısının iki olmasını beklemez ve ilgili etmenleri sisteme yollar. Yöntem_4'de ise, Mobile Agent'lar merkez tarafından değil de konakçılardaki Static Agent'lar tarafından merkezle koordineli olarak yaratılırlar. Yöntem_5'te, Mobile Agent'lar sızma-şüphelerini MainAgent ve diğer konakçılarla koordine kurmaksızın

belirleyen Static Agent'lar tarafından yaratılmakta ve yollanmaktadır. MainAgent hiç kullanılmadığından bu yöntem tam-dağıtık bir yöntemdir. Mobile Agent dağıtık sızma kararını verebilmek için bütün ağı rastgele bir sırada dolaşmaktadır. Yöntem_6'da, Yöntem_2'nin kısa ortalama belirleme süresi avantajı ile Yöntem_5'in tam-dağıtık yapısı birleştirilerek yüksek güvenirlik ve kısa belirleme süresi sahibi bir sistem geliştirilmiştir. Bu sistem, normalde mod-1'de (Yöntem_2) çalışırken, merkezi ünite kullanılamaz olduğunda mod-2'ye (Yöntem_5) geçmektedir. Mobile Agent'ların birbirleriyle ve diğer etmenlerle haberleşirken güvenli bir asıllama yaptıkları ve AlarmAgent ile MainAgent'ın güvenli bir alanda kurulu oldukları varsayılmaktadır.

Veri setinde yer alan DDoS saldırısının ilk üç aşaması doğru ve hızlı olarak belirlenerek, son iki aşaması gerçekleşmeden gerekli tedbirlerin alınması için güvenlik yöneticisinin uyarılması hedeflenmiştir. Yapılan yirmi adet koşturma sonucunda her bir yöntemin anılan sızma aşamalarını ortalama belirleme süreleri, ağdaki yük ve güvenirlik özellikleri ayrı ayrı değerlendirilmiştir. Gezgin etmenlerin kullanılmasının ağ yükünü artırdığı gözlenmiştir; fakat bu gezgin etmenlerin dağıtık olarak yaratılması sistemin merkezi üniteden olan bağımsızlığını da artırmaktadır. İmza-temelli bir SBS olan Snort'un geliştirdiği alarm dosyalarının kullanıldığı bu bölümde, mesajların büyüklükleri gezgin etmenlerinkinden küçüktür; fakat bir anomali-temelli SBS'inde bunun tam tersi durum oluşabilmektedir. Yöntem_4, merkezi üniteden tam-bağımsız olması ve düşük ağ yüküne sahip olması nedeniyle tezin bu aşaması için en iyi yöntem olarak değerlendirilmiştir. Bütün yöntemlerin çok yaklaşık "sızma belirleme süresi" değerleri olduğu için, bu süreler üçüncü bir kriter olarak kullanılmamıştır.

İkinci aşamada, SBS yapısı ve biyolojik bağışıklık sistemi arasındaki benzerlikten dolayı anomali-temelli sızma belirleme yöntemi olarak yapay bağışıklık sistemi kullanılmıştır. MIT DARPA 2000 LLS_DDOS 1.0 verisetindeki DDoS saldırılarını belirlerken daha iyi doğru ve yanlış pozitif oranları elde etmek amacıyla çok-amaçlı evrimsel algoritmadan esinlenen bir yapay bağışıklık sistemi olan jREMISA çalışmasına yeni geliştirmeler eklendi. Bu geliştirmeler: r-sürekli değerlendirme yönteminin eklenmesi, Negatif Seleksiyon ve Klonlama Seleksiyon'da değişiklikler yapılması, genel konsept korunurken ikinci hedefin yeniden tanımlanması olarak özetlenebilir.

Geliştirilmiş-jREMISA'nın gelen bir paketi inceleme süresi ortalama 2 milisaniye olarak ölçülmüştür. Ağ analiz aracı kullanılarak internetten bir video indirirken dahi paketlerin minimum 5 milisaniye aralıklı olarak (~200 paket/sn) geldiği düşünülürse, g-jREMISA'nın gelen trafiği yavaşlatmayacağı söylenebilir. Adaptif öğrenme modülü trafiğin akışını engellemeden paralel olarak çalışabilmektedir.

Geliştirilmiş-jREMISA üzerinde daha iyi doğru ve yanlış pozitif sonuçlar verecek en iyi parametre grubunu bulabilmek için benzerlik eşiği değerleri, r-sürekli değerleri ve birincil popülasyon büyüklüklerinden oluşan parametrelerin değişik ayarlamaları ile üç farklı test yapıldı. En sonunda, orjinal ve geliştirilmiş-jREMISA, önceden belirlenen en iyi parametre grubu kullanılarak karşılaştırıldı. Gelişimi görebilmek için, testler eşik değerleri değiştirilerek gerçekleştirilmiştir. Geliştirilen algoritmaya ait olan %100 doğru pozitif oranı ve %0 yanlış pozitif oranı, bir anomali sızma belirleme sistemi olarak kaydadeğer bir başarıdır.

Daha sonra, geliştirilmiş-jREMISA'nın performansını diğer benzer çalışmalarla karşılaştırabilmek için literatürde yaygın olarak kullanılan 1999 DARPA SBS veriseti günlerinin farklı bileşimleri kullanılarak iki çeşit deney yapıldı. Birinci tip deneylerde eğitim için kullanılan hafta-1'in günleri ile test için kullanılan hafta-4'ün günleri aynı olmasına rağmen bu günlerde bambaşka bir trafik yer almaktadır. İkinci tip deneylerde, daha gerçekçi bir test ortamını simule edebilmek için eğitim ve test aşamalarında hafta-1 ve hafta-4'ün farklı günleri kullanılmıştır. Her iki tip deneyde de 5 gün üzerinden ortalama alındığında yaklaşık %100 doğru pozitif oranı ve yaklaşık %0 yanlış pozitif oranı başarısı elde edilmiştir. Aynı veriseti üzerinde yapılan diğer çalışmalarla karşılaştırıldığında, geliştirilmiş-jREMISA'nın hepsinden daha iyi TP ve FP değerleri olduğunu gözlemlenmiştir.

Son olarak, ayrı ayrı geliştirilen bu iki yapı birleştirilerek tez çalışmasında hedeflenen sistem gerçeklenmiştir. Dağıtık sistemde her bir konakçıda yer alan imza-temelli bir SBS olan Snort tarafından oluşturulan sızma-şüphesi alarmlarının yerini anomali-temelli SBS olan geliştirilmiş-jREMISA'nın oluşturduğu alarmlar aldı. İmza-temelli kontrollere göre geliştirilen yöntemlerin üçünden benzer nedenlerle vazgeçilerek Yöntem_1 "Merkezi yöntem", Yöntem_5 "Tam-dağıtık yöntem" ve Yöntem_6 "Modüler-dağıtık yöntem" olarak yeniden düzenlendi. Mobile Agent'lar diğer konakçıların alarm dosyalarını sorgularken benzer bir anomali-tipi sızma-şüphesi paketini bulduğu konakçının durumunu sırasıyla DDoS-1, DDoS-2 ve

DDoS-3 yapmakta ve her seferinde konakçılara bir zararlı yazılımın yüklenip yüklenmediğini kontrol etmektedir. Güvenlik yöneticisi, bütün bu saldırı aşamalarından AlarmAgent aracılığıyla haberdar edilmektedir.

Bir Mobile Agent'ın ortalama boyutu 100 KB, iki konakçı arasında dolaşma süresi 100 msn olarak ölçülmüştür. Verisetindeki senaryo gereği 20 farklı konakçı ve bir güvenlik konsolü yaratılmış olup, 1000 kadar konakçı barındıran geniş bir ağ kullanılması durumunda dahi; en kötü durumda, MA dağıtık saldırı yapılan birinci konakçıyı birinci, ikinci konakçıyı sonuncu sırada dolaşacaktır. Bu durumda 20 konakçıda gecikme 2 sn, 1000 konakçıda 100 sn olacaktır. Sistemi geniş ağlar için daha ölçeklenebilir hale getirmek için, 50 veya 100'er konakçıdan oluşan konakçı gruplarına aynı görevli Mobile Agent'lar aynı anda gönderilebilir. Şüpheli bir saldırı kararına varan Mobile Agent "broadcasting" mesajı vasıtasıyla diğer gruplardaki aynı görevli Mobile Agent'ları sonlandırabilir.

İlk aşamada simule-gerçek zamanlı ortamda yapılan testler, birleşik sistem için de yapılmış ve oldukça başarılı sonuçlar alınmıştır. Yöntemler, yukarda belirtilen üç kritere göre değerlendirilmiş ama kendi aralarında doğrudan bir kıyaslama yapılarak en iyisinin seçilmesi işlemine gerek duyulmamıştır. Çünkü bu üç kriter, sistem geliştiriciler veya son kullanıcılar açısından farklı durumlarda farklı önceliklere sahip olabilirler. Literatürde "DDoS saldırılarının dağıtık olarak belirlenmesi", " Gezgin etmenler kullanarak sızmaların dağıtık olarak belirlenmesi" ve "Anomali-temelli SBS" konularında çalışmalar olmasına rağmen, bu tez çalışması, DDoS saldırıları öncesi hareketleri orta katmanda dağıtık olarak belirlemeye çalışan ilk çalışmadır.

Bu çalışmada, imza-temelli ve anomali-temelli SBS'ler ayrı ayrı ele alınmış olup aynı yöntemler üzerinde ayrı testler yapılarak sonuçlar değerlendirilmiştir. Sızma-şüphesi alarmları oluşturulurken, sadece konakçılara gelen ağ trafiği dikkate alınmış olup konakçılardaki işletim sistemi, registry kayıtları, log kayıtları ile ilgilenilmemiştir. DDoS saldırılarını ve öncesindeki dağıtık hareketleri her türlü ortamda başarıyla belirleyebilmek için imza-temelli ve anomali-temelli dSBS'ler; ağ-temelli ve konakçı-temelli dSBS'ler beraber kullanılmalıdır.

Etmenlerin ve mesajların güvenliği ileriki çalışmalarda değerlendirilmelidir. Bu çalışmada haberleşmelerin güvenlikli bir asıllama ile yapıldığı varsayılmaktadır. Çalışmanın yapısını güçlendirmek için diğer yaygın sızma belirleme verisetleri de

sistem üzerinde test edilebilir. Güncel trafik ve saldırı senaryolarını içeren özgün bir veriseti hazırlanabilir. Kurulan bütünleşik sistemin ilk aşamasında g-jREMISA yerine adaptif öğrenme özelliği olan başka bir SBS algoritması denenebilir. Bütünleşik sistem, etkin bir arayüz ile beraber konakçılara kolaylıkla kurulabilecek bir paket program haline getirilebilir. Son olarak, bu SBS'nin, uygun bir doğruluk kümesi ile beraber gerçek-zamanlı ağ trafiğinde de test edilmesi tavsiye edilmektedir.

KAYNAKLAR

[1] **Farmer, J.G.,** 2006. Information Security: The Nature and Structure of Intrusion Detection Systems, *Ph.D. dissertation*, Applied Management and Decision Sciences Faculty, Walden Univ., Minneapolis, MN.

[2] **Das, V. ve diğ.,** 2010. Network Intrusion Detection System Based on Machine Learning Algorithms, *International Journal of Computer Science & Information Technology (IJCSIT)*, DOI: 10.5121/ijcsit.2010.2613, Vol 2, No 6.

[3] **Kannadiga, P. ve Zulkernine, M.,** 2005. DIDMA: A Distributed Intrusion Detection System Using Mobile Agents, *Proceeding of the ACIS 6th International Conference on Software Engineering, Networking and Parallel/Distributed Computing (SNPD/SAWN)*, pp. 238-245, 23-25 May.

[4] **Chandler, J.A.,** 2003. Security in Cyberspace: Combatting Distributed Denial of Service Attacks, *University of Ottawa Law & Technology Journal*, Vol. 1, p. 231.

[5] **Haag, C.R., Lamont, G.B., Williams, P.D. ve Peterson G.L.,** 2007. An artificial immune system-inspired multiobjective evolutionary algorithm with application to the detection of distributed computer network intrusions, *GECCO '07: Genetic and evolutionary computation Conference*, London, UK.

[6] **Du, Y., Wang, H.-Q. ve Pang, Y.-G.,** 2004. IADSBS-Design of A Distributed Intrusion Detection System Based on Independent Agents, *Proceedings of International Conference on Intelligent Sensing and Information Processing*, pp. 254 – 257.

[7] **Crosbie, M. ve Spafford, G.,** 1995. Defending a Computer System using Autonomous Agents, *18th National Information Systems Security Conference.*

[8] **Burbeck, K.,** 2006. Adaptive Real-time Anomaly Detection for Safeguarding Critical Networks, *BSc Thesis,* Linköping University, Sweden.

[9] **Patel, A., Qassim, Q. ve Wills C.**, 2010. A Survey of Intrusion Detection and Prevention Systems, *Information Management &Computer Security Journal,* Vol. 18 No. 4, pp. 277-290, Emerald Group Publishing Limited.

[10] **Hamed, E.,** 2001. An agent-based intrusion detection system using fuzzy logic for computer system threat evaluation, *Ph.D. dissertation*, Graduate School of the University of Louisville.

[11] **Zheng, J. ve Hu, M.-Z.,** 2005. Intrusion Detection of DoS-DDoS and Probing Attacks for Web Services, WAIM, *LNCS 3739*, pp 333-344.

[12] **Kruegel, C. ve Toth, T.,** 2002. Distributed Pattern Detection for Intrusion Detection, *Network and Distributed System Security Symposium Conference (NDSS).*

[13]<*http://www.snort.org*>, Snort – the de facto standard for intrusion detection/prevention, alındığı tarih 13.03.2011.

[14] **Northcutt, S. ve Novak, J.,** 2003. Network Intrusion Detection, *New Riders Publishing.*

[15] **Smith, R.,** 2006. Correlating intrusion alerts with unsupervised learning, *Ph.D. dissertation*, Faculty of Engineering, University of Ottowa.

[16] **Magalhaes, A.,** 2010. Using Intelligent Mobile Agents to Dynamically Determine Itineraries with Time Constraints, Proceedings *of the - 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'2010).*

[17] **Foukia, N.,** 2005. IDReAM: Intrusion Detection and Response executed with Agent Mobility Architecture and Implementation, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05),* pp. 264-270.

[18] **Ramachandran, G. ve Hart, D.,** 2004. A P2P intrusion detection system based on mobile agents, *Proceedings of the 42nd Annual Southeast Regional Conference (ACM-SE 42),* pp. 185-190, April.

[19] **Chen, Y. ve Zhou L.,** 2004. An Innovative SBS immune System Model, *IEEE International Conference on Systems, Man & Cybernetics (SMC'04),* 4810-4814.

[20] **Lange, D. ve Oshima, M.,** 1998. Programming and deploying Java mobile agents with Aglets, *Addison Welsey.*

[21] **Anand, R. ve diğ.,** 2002. Design of the Ajanta System for Mobile Agent Programming , *Journal of Systems and Software*, May.

[22] **Wong, D. ve diğ.,** 1997. Concordia: An Infrastructure for Collaborating Mobile Agents, *In Mobile Agents: First International Workshop, Lecture Notes in Computer Science, Vol. 1219*, Springer-Verlag, Berlin,Germany.

[23] **Gray, R. ve diğ.,** 1998. D'Agents: Security in a multi-language, mobile agent system, In G.Vigna, editor, Mobile Agents and Security, pages 154-187, *Spring-Verlag.*

[24]<*http://www.recursionsw.com/Products/voyager.html*>, Recursion Software: Products: Voyager Edge: Solutions for intelligent mobile applications, rapid development, and social networking, alındığı tarih 17.06.2010.

[25] **Newman, H.B., Legrand, I.C. ve Bunn, J.J.**, 2001. A Distributed Agent-based Architecture for Dynamic Services, *Computing in High Energy Physics (CHEP).*

[26] **Jazayeri, M. ve Lugmayr, W.,** 2000. Gypsy: A Component-based Mobile Agent System, *Eighth Euromicro Workshop on Parallel and Distributed Processing (EURO-PDP 2000),* Rhodos Greece, January 19-21.

[27] **Binder, W.,** 2001. Design and Implementation of the J-SEAL2 Mobile Agent Kernel, *Proceedings of the 2001 Symposium on Applications and the Internet (SAINT 2001)*, p.35, January 08-12.

[28] **Undercoffer, J., Perich, F. ve Nicholas, C.,** 2002. SHOMAR: An Open Architecture for Distributed Intrusion Detection Services, *Technical report*, University of Maryland, Baltimore County.

[29] **Bellifemine, F. ve diğ.,** 2003. JADE - A white paper, EXP in search of innovation - *Special Issue on JADE, TILAB Journal*.

[30] **Bellifemine, F., Caire, G. ve Greenwood, D.**, 2007. Developing Multi Agent Systems with Jade, England, *John Wiley and Sons Ltd.*, pp. 1-34.

[31] **Ko C. ve diğ.,** 1993. Analysis of an Algorithm for Distributed Recognition and Accountability, *Proceeding of 1stACM Conf. On Computer and Communications Security*, Fairfax, VA, (154—164).

[32*]<http://www.team-cymru.org/ReadingRoom/Whitepapers/2010/ddos-basics.pdf>*, DDoS Basics, alındığı tarih 04.03.2011.

[33] **Loukas, G. ve Öke, G.,** 2009. Protection Against Denial of Service Attacks: A Survey, *The Computer Journal*, Vol. 53, No. 7, Oxford University Press.

[34] **McPherson, D. ve diğ.,** 2009. *Worldwide Infrastructure Security Report*, Arbor Networks.

[35]<*http://www.tech-mavens.com/synflood.htm*>, Oliver, R., Countering SYN Flood Denial of Service Attacks, alındığı tarih 24.02.2011.

[36] **Zhong, R. ve Yue, G.,** 2010. DDoS Detection System Based on Data Mining, *Proceedings of the Second International Symposium on Networking and Network Security (ISNNS '10)*, Jinggangshan, P. R. China, pp. 62-65.

[37] **Park, H., Kwak, N-S. ve Lee, J.,** 2009. A method of multiobjective optimization using a genetic algorithm and an artificial immune system, *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 223: 1243, DOI: 10.1243/09544062JMES1313.

[38] **Bıyıklıoğlu, O.,** 2004. Use of artificial immune systems for network intrusion detection, *Master thesis*, Computer Engineering Department, Istanbul Technical University, Istanbul, Turkey.

[39] **Coello, C.C.A. ve Lamont, G.B.,** 2005. An Introduction to Multi-Objective Evolutionary Algorithms and Their Applications, in (*C.A. Coello Coello, Ed.*) Applications of Multi- Objective Evolutionary Algorithms.

[40] **Al-Enezi, J.R. ve diğ.,** 2010. Artificial Immune Systems – Models, Algorithms And Applications, *International Journal of Research and Review in Applied Sciences,* 3 (2*),* Academic Research Publishing Agency Press.

[41] **Visconti, A. ve Tahayori, H.,** 2011. Artificial immune system based on interval type-2 fuzzy set paradigm, *Applied Soft Computing Journal*, 4055–4063, doi:10.1016/j.asoc.2010.12.011.

[42] **Aickelin, U. ve Dasgupta D.,** 2005. Artificial Immune Systems Tutorial, Chapter 13 of Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies, *Eds. E. Burke, G. Kendall, Springer.*

[43] **Chu, Y., Li, J. ve Yang Y.,** 2005. The Architecture of the Large-scale Distributed Intrusion Detection System. *Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05),* 130-133.

[44] **Zhang, Z., Luo, W. ve Wang, X.,** 2005. Designing Abstract Immune Mobile Agents for Distributed Intrusion Detection, *International Conference on Neural Networks and Brain (ICNN&B '05),* 748-753.

[45] **Bernardes, M.C. ve Moreira, E. dos S.,** 2000. Implementation of an Intrusion Detection System Based on Mobile Agents, *Proceedings of the International Symposium on Software Engineering for Parallel and Distributed Systems (PDSE 2000),* pp. 158-164.

[46] **Zhong, S.-C. ve diğ.,** 2003. Safe Mobile Agent System For Distributed Intrusion Detection, *Proceedings of The Second International Conference on Machine Learning and Cybernetics*, pp. 2009-2014, November.

[47] **Asaka, M. ve diğ.,** 1999. A Method of Tracing Intruders by Use of Mobile Agents, *Proceedings of the 9th Annual Conference of the Internet Society (INET'99),* June.

[48] **Eskin, E. ve diğ.,** 2000. Adaptive Model Generation for Intrusion Detection Systems, *In Workshop on Intrusion Detection and Prevention, 7th ACM Conference on Computer Security.*

[49] **Hu, W., Liao, Y. ve Vemuri, V.R.,** 2003. Robust Support Vector Machines for Anomaly Detection in Computer Security, *International Conference on Machine Learning and Applications (ICMLA'03).* Los Angeles, California, Haziran.

[50] **Ertoz L. ve diğ.,** 2004. MINDS - Minnesota Intrusion Detection System, *Data Mining - Next Generation Challenges and Future Directions*, Chapter 3, MIT Press.

[51] **Vokorokos, L., Anton, B. ve Martin, C.,** 2006. Intrusion Detection System Using Self Organizing Map, *Acta Electrotechnica et Informatica*, No 1, Vol 6.

[52] **Wang, K., Stolfo, S.J.,** 2004. Anomalous Payload-based Network Intrusion Detection, *7th International Symposium on Recent Advances in Intrusion Detection (RAID),* 203—222.

[53] **Bolzoni, D. ve diğ.,** 2006. POSEIDON: a 2-tier Anomaly-based Network Intrusion Detection System, *Fourth IEEE International Workshop on Information Assurance (IWIA'06),* 144-156.

[54] **Kohonen, T.,** 2001. Self-Organizing Maps. *Springer Series in Information Sciences*, Vol. 30, Springer, Third Extended Edition.

[55] **Mahoney, M.V. ve Chan, P.K.,** 2001. PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic, Florida Tech., *Technical report,* 2001-04.

[56] **Gavrilis, D. ve Dermatas, E.,** 2005. Real-time detection of distributed denial-of-service attacks using RBF networks and statistical features, *Computer Networks and ISDN Systems, Elsevier,* Vol 48, Issue 2.

[57] **Das, K.,** 2001. Protocol Anomaly Detection for Network-based Intrusion Detection, *GSEC Practical Assignment Version 1.2f.*

[58] **Szmit, M. ve diğ.,** 2007. Traffic Anomaly Detection with Snort, Chapter in *Information Systems Architecture and Technology, Information Systems and Computer Communication Networks,* Wydawnictwo Politechniki Wrocławskiej, Wrocław.

[59] **Aydin, M.A., Zaim, A.H. ve Ceylan, K.G.,** 2009. A hybrid intrusion detection system design for computer network security, *Computers and Electrical Eng. Journal,* 35 (3), 517-526.

[60] **Kayacik, G. H. ve Zincir-Heywood, A.N.,** 2003. Using Intrusion Detection Systems with a Firewall: Evaluation on DARPA 99 Dataset, *NIMS Technical Report,* #062003.

[61] **Takeda, K. ve Takefuji, Y.,** 2001. Pakemon – A Rule Based Network Intrusion Detection System, *Int. Journal of Knowledge-Based Intelligent Engineering Systems,* Vol. 5, No. 4, 240-246.

[62]<*http://www.cisco.com/en/US/docs/ios/12_0t/12_0t5/feature/guide/ios_SBS.html*>, Cisco IOS Firewall Intrusion Detection System, alındığı tarih 05.07.2010.

[63] **Gaddam, S.R., Phoha, V.V. ve Balagani, K.S.,** 2007. A Novel Method for Supervised Anomaly Detection by Cascading K-Means Clustering and ID3 Decision Tree Learning Methods, *IEEE Trans. on Knowledge and Data Engineering,* v.19 n.3, 345—354.

[64] **Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.,** 2007. Evolutionary Algorithms for Solving Multi-Objective Problems, Genetic and Evolutionary Computation, *Springer, 2nd edition.*

[65] **Edge, K., Lamont, G., Raines, R.,** 2006. A Retrovirus Inspired Algorithm for Virus Detection & Optimization, *GECCO '06,* July 8-12.

[66] **Coello, C., Cortés, N.,** 2005. Solving Multiobjective Optimization Problems Using an Artificial Immune System, Genetic Programming and Evolvable Machines, Vol. 6, pp.163-190.

[67] **Mahoney, M.V.,** 2003. Network traffic anomaly detection based on packet bytes, *ACM Symposium on Applied Computing (SAC).*

[68]<*http://www.winsnort.com*>, WinIDS Installation Guide, alındığı tarih 26.01.2010.

[69] **Mahoney, M.V. ve Chan, P.K.,** 2003. An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection, *Proceedings of Recent Advances in Intrusion Detection: 6th International Symposium (RAID 2003),* pp. 220-239, Pittsburgh, PA, USA, 8- 10 September.

[70] **Brugger, S.T ve Chow, J.,** 2007. An Assessment of the DARPA SBS Evaluation Dataset Using Snort, *UC Davis Technical Report*, CSE-2007-1, Davis, CA, 6 January.

[71]<*http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000/LLS_DDOS_1.0.html*>, MIT Lincoln Laboratory, Information Systems Technology, alındığı tarih 12.04.2010.

[72]<*http://www.ethereal.com*>, Ethereal: Open-source network protocol analyzer, alındığı tarih 14.03.2011.

[73] **Haines, J.W. ve diğ.,** 2001. 1999 DARPA intrusion detection evaluation: design and procedures, *MIT Lincoln Laboratory Technical Report*, TR-1062, Massachusetts

[74] **Neumann, P. ve Porras P.,** 1999. Experience with EMERALD to DATE, *1st USENIX Workshop on Intrusion Detection and Network Monitoring*, California, 73-80.

[75] **Vigna, G., Eckmann, S.T. ve Kemmerer, R.A.,** 2000. *The STAT Tool Suite*, DARPA Information Survivability Conference and Exposition (DISCEX).

[76] **Barbara, D. ve diğ.,** 2001. ADAM: Detecting Intrusions by Data Mining, *IEEE SMC Information Assurance Workshop*, West Point, NY.

[77] **Tyson, W.M.** 2001. DERBI: Diagnosis, Explanation and Recovery from Computer Break-ins, *Final Report*, Artificial Intelligence Center, SRI Int., DARPA Project F30602-96-C-0295

[78]<*http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000/LLS_DDOS_2.0.2.html*>, MIT Lincoln Laboratory, Information Systems Technology, alındığı tarih 10.08.2010.

ÖZGEÇMİŞ

Ad Soyad: Uğur Akyazı

Doğum Yeri ve Tarihi: Şabanözü/Çankırı – 02.05.1978

Adres: Hava Dış Lojmanları 10/2 Yeşilyurt Bakırköy İstanbul

Lisans Üniversite: Hava Harp Okulu

Yayın Listesi:

- **Akyazı U.**, Uyar A.Ş., 2011: Distributed Detection of DDoS Attacks During the Intermediate Phase Through Mobile Agents, *Computing and Informatics Journal*, ISSN 1335-9150, Slovak University Press, Bratislava, (Baskı aşamasında).

- **Akyazı U.**, Uyar A.Ş., 2010: A Hybrid Multiobjective Evolutionary Algorithm for Anomaly Intrusion Detection. *International Symposium on Distributed Computing and Artificial Intelligence (DCAI'10)*, September 7-10, Valencia, Spain.

- **Akyazı U.**, Uyar A.Ş., 2010: Detection of DDoS Attacks via an Artificial Immune System-Inspired Multiobjective Evolutionary Algorithm, *EvoCOMNET: 7th European Event on the Application of Nature-inspired Techniques for Telecommunication Networks and other Parallel and Distributed Systems* , April 7-9, İstanbul, Turkey, LNCS vol. 6025, Springer.

- **Akyazı U.**, Uyar A.Ş., 2008: Distributed Intrusion Detection using Mobile Agents against DDos Attacks, *23rd International Symposium on Computer and Information Sciences (ISCIS)*, October 27-29, İstanbul, Turkey, IEEE, ISBN: 978-1-4244-2880-9, DOI:10.1109/ISCIS.2008.4717920.