HAREKETLİ ETMENLER İÇİN GÜVENLİ BİR ÇALIŞMA ORTAMI

Suat UĞURLU[1] **Nadia ERDOĞAN[2]**

[1,2]Bilgisayar Mühendisliği Bölümü
Elektrik-Elektronik Fakültesi
İstanbul Teknik Üniversitesi, 34390, Maslak, İstanbul
[1]e-posta: suat@suatugurlu.com [2]e-posta: erdogan@cs.itu.edu.tr

Anahtar sözcükler:Hareketli Etmenler, Hareketli Etmen Sistemlerinde Güvenlik

ÖZET

Bilgisayar sistemlerinin gelişim sürecine baktığımızda, yazılım metodolojilerinin gün geçtikçe merkezi çalışma modellerinden, dağıtık çalışma modellerine doğru evrim geçirmekte olduğunu görmekteyiz. Mesajlaşma ile başlayan dağıtık çalışma yöntemleri, uzaktan yordam çağırma ile devam etmiş ve en sonunda ağ ortamında farklı bilgisayarlardaki nesnelerin kullanılmasına izin veren yöntemler ile süregelmiştir. Bu gelişimin son aşamasında, nesnelerin ağ ortamında hareketliliğini sağlayan ve heterojen ortamlarda çalışabilen hareketli etmenleri görüyoruz. Hareketli etmen mimarisi istemci-sunucu çalışma modeline karşın dağıtık işlemeye farklı bir yaklaşım sunmaktadır. Kodun hareketliliğine dayanan hareketli etmen sistemlerinde, güvenlik düşünülmesi gereken önemli bir unsurdur. Hareketli olan kod, yaşam süresince bir çok risk altındadır. Bu bildiride, güvenli bir hareketli etmen mimarisi olan "Güvenli Etmen Sistemi" (GES)'in tasarım ve gerçekleme ayrıntıları incelenmektedir. GES önerdiği yeni sarmalanmış etmen modeli ile etmeni çevresindeki tehlikelerden korur. GES aynı zamanda etmenler arası güvenli iletişime, güvenli etmen göçlerine de olanak sağlamakta ve yetkisiz erişimleri izlemek ve denetlemek üzere güvenlik politikaları kullanmaktadır.

1. GİRİŞ

Dağıtık veri işleme yöntemleri mesajlaşma ile başlamış ve gelişimine uzaktan yordam çağırma ile devam etmiştir (RPC). Nesneye yönelik programlama dillerindeki gelişmelerle birlikte ağdaki farklı bilgisayarlardaki nesne metodlarının kullanımına izin veren (CORBA [1], DCOM[1], RMI[1]) daha karmaşık yöntemler tasarlanmış ve uygulanmıştır.

Bir sonraki evrede ise istemci-sunucu modeline alternatif olan ve kod hareketliliğine dayanan sistemleri görmekteyiz. Yakın zamanda kod hareketliliğine dayanan çeşitli sistemler önerilmiş ve bazıları uygulama alanı da bulmuştur. İstek anında kod (code on demand), uzaktan değerlendirme (remote evaluation) ve hareketli etmenler, kod ve durum bilgisinin ağdaki farklı bilgisayarlar arasında taşınmasının söz konusu olduğu sistemlere örnek olarak verilebilir.

Genel olarak hareketli bir etmen; otonom, amaca yönelik çalışan, kullanıcı adına davranan ve amacına ulaşmak için ağdaki düğümler arasında hareket etme yeteneğine sahip yazılım parçaları olarak tanımlanabilir[2]. Etmen, davranışlarını belirleyen bir kod parçası ve durumunu saklayan veri kısmından oluşur. Etmen verisini (durumu) durağan ve dinamik olarak da ikiye ayırmak mümkün olabilir. Durağan veri etmen yaşam süresi boyunca değişmez; dinamik veri ise hesaplamalar ve göçler sonucu yenilenen değişken bölümdür. Etmenin ilk yaratıldığı yere etmenin evi denir ve etmen yaşam süresi boyunca diğer etmenler ile haberleşme, göç etme, düğüm üzerinde kaynak kullanma ve klon yaratma gibi çeşitli aktivitelerde bulunabilir[3]. Hareketli etmen tabanlı sistemler, daha az ağ trafiği ve ağ gecikmesi oluşturma, hata hoşgörülü olma, heterojen ortamlarda çalışabilme ve ölçeklenebilirlik gibi avantajlarından dolayı istemci-sunucu modeline göre daha tercih edilebilir bir alt yapı sunarlar. Ayrıca, etmen tabanlı programlama modeli, karmaşık dağıtık sistemlerin tasarımına da olanak vermektedir. Bugüne kadar çeşitli hareketli etmen sistemleri önerilmiş ve geliştirilmiş, bazıları ise diğerlerinden daha çok ilgi görmüştür. Telescript[4], Tacoma[5], AgentTcl[6], Aglets[7], Voyager[8], Concordia[9] ve Ajanta[10] en popüler hareketli etmen sistemleri olarak kullanım alanı bulmuşlardır.

Bütün avantajlarına rağmen hareketli etmen sistemlerinde bir çok güvenlik riski bulunmaktadır [11],[12]. Hem hareketli olan etmenler, hem de etmenlere çalışma ortamı sunan düğümler bu güvenlik tehlikelerine karşı korunmalıdır. Hareketli etmen sistemlerinde karşılaşılabilecek atakları üç grup altında toplayabiliriz: Düğümlerden etmenlere yönelik ataklar, etmenlerden düğümlere yönelik ataklar ve etmenlerden etmenlere yönelik ataklar. Düğüm ve etmenlerin maruz kalabileceği çok daha karmaşık atak çeşitleri de söz konusu olabilmektedir.

- **Düğümlerden etmenlere yönelik ataklar:** Etmeni üzerinde barındıran düğüm etmene ait gizli verileri çalabilir, kodunu değiştirebilir. Düğüm ayrıca etmeni yanıltmaya yönelik yanlış bilgilendirmede bulunarak ya da etmene yetersiz kaynak sunarak (işlemci, disk, ağ, vb.) etmenin gereğinden fazla zaman harcamasına veya yanlış sonuçlar elde etmesine neden olabilir.
- **Etmenlerden düğümlere yönelik ataklar:** Zararlı etmenler üzerinde çalıştığı düğüme ait kullanıcı adı-şifre gibi gizli bilgileri çalabilir, ya da kullanıcıyı rahatsız edici davranışlarda bulunabilir. Düğüm üzerinde aşırı kaynak kullanımı ile hizmet kıtlığına (DOS) sebep olabilir.
- **Etmenler arası ataklar:** Zararlı bir etmen diğer bir etmenin yerine geçmek isteyebilir, diğer etmenlere ait kodu veya verileri değiştirebilir, veya etmenin görevini yerine getirmesini engellemek için yanlış bilgilendirmede bulunabilir.

Güvenli bit hareketli etmen sistemi, etmen haberleşmesi veya göçü gibi temel etmen aktivitelerini yerine getirmenin yanısıra, programcıya ek yük getirmeden, bu atakların bir çoğuna karşı koyabilecek güvenlik mekanizmalarını da içermelidir. Halihazırdaki sistemlerin, bu atak çeşitlerinden çok az bir kısmına karşı gerekli fonksiyonları içerdiğini görmekteyiz. Bununla beraber, bir çok araştırmacı farklı türden ataklara karşı önlem olabilecek çeşitli önerilerde bulunmuştur [13], [14], [15], [16], [17], [18], [19]. Ancak, bu çözümlerin bir çoğu pratikte kullanılan bir hareketli etmen sistemine uyarlanamamıştır. Önerilen yöntemlerin büyük bir kısmı etmen ve düğümü korumaya yönelik sınırlı güvenlik mekanizmaları içerirken, bir çoğu da programcıya ek yük getirmektedir. Uygulama güçlüğü nedeniyle hizmet kıtlığı ataklarına karşı etkin bir çözüm geliştirilememiştir. Ancak, gelişmiş bir hareketli etmen sistemi en azından etmen aktivitelerini izleyen ve kaydeden fonksiyonları içermeli, bu izlerden ileride gerçekleşebilecek etmen ataklarına karşı diğer etmenleri ve düğümleri korumaya yönelik çıkarımlar yapabilmelidir. Ayrıca, politika tabanlı dinamik bir güvenlik yönetiminin bir çok sistemde eksik olduğu görülmektedir.

Bu bildiride yeni ve güvenli bir hareketli etmen mimarisi olan GES anlatılmaktadır. GES, sarmalanmış etmen modeli ile etmeni bulunduğu ortamdan yalıtır. GES aynı zamanda etmen iletişimi ve göçleri için güvenli mekanizmalar içerir. Yetkisiz erişimleri engellemek ve izlemek için güvenlik politikaları barındırır.

2. GES GÜVENLİK MODELİ

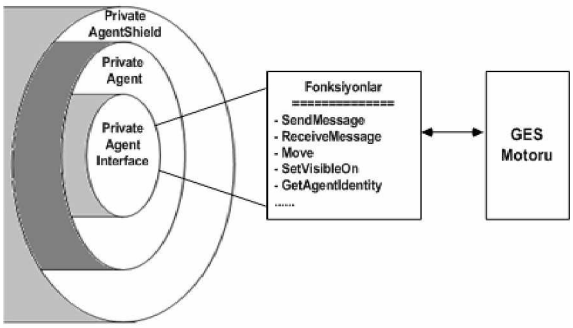
Güvenli bir sistemin tipik parçaları ; *varlıklar, güvenlik politikaları* ve *koruma alanları*dır[20]. Varlıklar, bilgiye ulaşan birimlerdir ve kaynaklara erişim güvenlik politikaları ile denetlenir; varlıklar kullanıcılar, sunucular, veya IP adresleri olabilir. Güvenlik politikaları, varlıklara, kaynaklara ve düğümler tarafından sunulan hizmetlere erişim için gerekli olan hakları atar. Güvenlik politikaları aynı zamanda varlıklar arası haberleşmeleri de kontrol eder[20].

Hareketli etmen sistemlerinde etmenler, yeterli güvenlik önlemleri alınmadan son kullanıcılar gibi düşünülemezler ve yaşamları boyunca çeşitli güvenlik tekditleri ile karşı karşıyadırlar. Diğer etmen sistemlerinde olduğu gibi GES, her etmeni ayrı bir varlık gibi ele alır ve etmenleri birbirlerinden soyutlayacak güvenlik mekanizmalarını sunar. GES, etmenleri soyutlama yöntemi ve güvenlik politikalarının kullanımı ile diğer etmen sistemlerinden ayrılmaktadır. GES, etmenleri ve düğümleri korumayı sağlamakla birlikte güvenli etmen haberleşmesi ve güvenli etmen göçünü de desteklemektedir. GES etmenlerin bellek kullanımı sınırlayabilir, güvenlik politikalarını uygular, ve etmenin yaratılmasından yok edilmesine kadar olan bütün etmen aktivitelerini izler ve kaydeder.

Mimarinin ana bileşeni her düğüm üzerinde çalışan ve etmenlere ev sahipliği yapan GES sunucusudur. GES sunucuları Şekil 1. de görüldüğü gibi sarmalanmış etmen modeli ile etmene güvenli bir çalışma ortamı sunmaktadır. GES sunucusu bir etmen yaratıldığı anda etmeni çevreleyen özel (pivate) bir *AgentShield* nesnesi yaratır. Etmen, yaratılan bu özel nesne içinde yeniden özel bir nesne olarak tanımlanmıştır. Böylece etmen kod ve verisi bütünüyle diğer etmenlerin doğrudan erişimlerine karşı korunmuş olur. Mimari birden fazla etmenin aynı anda düğüm üzerinde çalışmasına olanak vermektedir. Her etmen ayrı bir *iplik (thread)* olarak çalışmaktadır, dolayısı ile bütün etmenler aynı bellek alanını paylaşırlar. Sarmalanmış etmen modeli aynı bellek alanında çalışan etmenlerin birbirlerine doğrudan ulaşmalarını engellemeyi garanti etmektedir. Etmenin çevresi ile etkileşimini *AgentInterface* adındaki, önceden tanımlı sınırlı sayıda etkileşim fonksiyonu içeren özel bir nesne sağlar. Bu nesne etmenin GES sunucular ve diğer etmenler ile etkileşim için kullanabileceği tek araçtır.

GES, güvenlik mekanizmalarını gerçeklerken şifreleme tekniklerinden de faydalanır. Her GES sunucusu kimliğini belirleyen ve gizlenmesi gereken verileri şifrelemek için kullandığı özel-açık anahtar ikilisi barındırır. GES etmenlerine ait kod ve durum bilgisi yaşamları boyunca şifreli olarak tutulmaktadır. Etmen kodunun açık hale getirildiği tek an etmenin bellekte aktif duruma getirildiği andır. Yani etmen bellekte bulunduğu an hariç, sistemde üçüncü birimler için bir karakutudur. GES etmenleri mesajlar ile haberleşirler.

Sistem, *AgentInterface* nesnesinin sunduğu metodlar aracılığı ile asenkron haberleşmeye olanak vermektedir. Bütün etmen-etmen haberleşmeleri SSL ile şifreli olarak gerçeklenir.



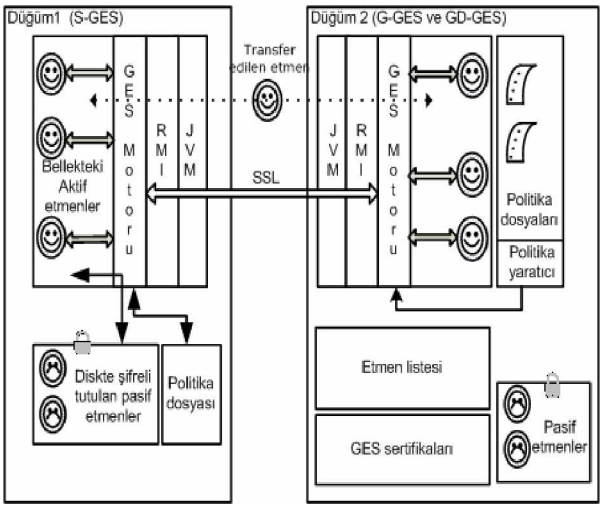
Şekil 1. Etmen soyutlama modeli

GES aynı zamanda etmenleri ağ üzerinde güvenli bir şekilde göç etmesine de destek vermektedir. Hem etmen kodu hem de etmen verisi iletim sırasında şifrelenmektedir ve doğru anahtarı bilen alıcı GES sunucu tarafında şifre çözülmektedir.

GES, etmenlerin belirli aktivitelerini denetlemek için politika tabanlı yetkilendirme mekanizması kullanır. Etmen haberleşmesi, göçü, düğüm üzerindeki diskten okuma ve yazma, ağ kaynaklarının kullanımı denetim altına alınabilen başlıca aktivitelerdir. GES sunucular etmen aktivitelerini kaydederler. Bu izler etmen davranışını incelemek için kullanılabilir. Hatta, ilk aşamada sezilmesi çok zor olan atakların bu izler aracılığı ile bulunması da söz konusu olabilir. GES sunucular gerek gördükleri anda bir etmenin çalışmasını sona erdirebilir.

3. GES MİMARİSİ

Güvenlik, sistem tasarımı aşamasında düşünülmesi gereken bir olgu olduğu için, var olan bir hareketli etmen sistemine güvenlik fonksiyonlarını eklemek yerine tamamiyle yeniden bir hareketli etmen sisteminin geliştirilmesi öngörülmüştür. Güvenlik, hareketlilik ve dağıtık işlemeye yönelik desteği nedeniyle sistem Java dili üzerinde geliştirilmiştir. Şekil 2 GES mimarisini göstermektedir. Sistemin ana bileşeni etmen yaratılmasından, aktive edilmesinden, haberleşmesinden ve göç ettirilmesinden sorumlu olan GES sunucusudur. Bütün GES sistemi etmenler için çalışma ortamı sağlayan birden fazla GES sunucunun birlikte çalışmasından oluşur. Teorik olarak bir GES sunucu düğüm kaynakları yeterli olduğu sürece yeni bir etmene hizmet verebilir. GES, sarmalanmış etmen modeli ile etmeni bulunduğu ortamdan yalıtarak etmeni diğer etmenlere karşı korur. Aynı zamanda bütün etmen aktiviteleri GES sunucular kontrolünde gerçeklendiği için bir etmenin ne yapmaya çalıştığı saptanabilir ve yetkisiz etmen aktivitelerine izin verilmez. GES sunucular çevresi ile etkileşimi için etmene bir arayüz sunar. Etmen bu arayüz aracılığı ile diğer etmenler ile haberleşmek istediğini ya da yeni bir düğüm üzerine taşınmak istediğini bildirir. Arayüz sınırlı sayıda fonksiyon içerir ve etmenin GES sunucu ile haberleşmesi için tek yoldur.



Şekil 2. GES mimarisi

3.1 GES SUNUCUSU

Her GES sunucu üstlendiği görevin türüne göre üç farklı düzende çalışabilir: Standart Düzen, Gözleyici Düzeni ve Güvenlik Yöneticisi Düzeni. GES sunucunun hangi düzende çalışacağını kullanıcının grafik arayüz aracılığı ile tanımladığı konfigürasyon belirler. GES sunucu birden fazla düzende aynı anda çalışabilir. Sistemde tek bir Gözleyici ve Güvenlik yöneticisi bulunabilir.

Standard Düzen (S-GES): Standart GES sunucusu etmen yaratma, aktive-pasif etme, haberleşme ve etmen göçü gibi temel etmen aktivitelerini yerine getirir. Aynı zamanda Güvenlik Yöneticisinden aldığı güvenlik politikalarını göre etmen aktivitelerini denetler. GES sunucu üzerindeki kurallar bir politika dosyasında şifreli olarak disk üzerinde tutulur. Standart GES sunucunun diğer bir görevi o an üzerinde aktif durumda bulunan etmenler listesini Gözleyici düzende çalışan GES sunucuya iletmektir. Etmen aktivitelerinin kaydedilmesi Standart GES sunucunun sorumlu olduğu diğer bir iştir. İz dosyaları disk üzerinde tutulur ve istendiğinde incelenebilir. Bu kayıtlar kolaylıkla sezilmesi mümkün olmayan atakların bulunması veya hata ayıklamaya yönelik etmen aktvitelerinin izlenmesi açısından çok önemlidir.

Gözleyici Düzen (G-GES): Gözleyici düzende çalışan GES sunucu standart bir GES sunucunun sağladığı bütün fonksiyonları sağlar. Etmenler hareketli olduklarından, bir etmene ait yer bilgisi zamanla değişmektedir. Bu nedenle etmenler arası haberleşme öncelikle etmeni işaret eden bir referansın elde

edilmesiyle başlar. Gözleyici GES sunucu, sistemde o an aktif olan bütün etmenlere ait kimlik-yer veri tabanını tutar. Diğer bir etmenle haberleşmek isteyen etmen bu isteğini yerel GES sunucuya ilettiğinde yerel GES sunucu öncelikle hedef etmenin o anki yer bilgisini Gözleyici düzende çalışan GES sunucudan öğrenir ve bunu etmene iletir. Kaynak etmen bu bilgi ile hedef etmene mesaj gönderebilir. Her GES sunucu etmen aktif-pasif veya pasif-aktif durum geçişlerini Gözleyici GES sunucuya iletmektedir. Bu şekilde konumdan bağımsız etmen haberleşmesi gerçeklenmiş olur.

Güvenlik Denetleyicisi Düzen (GD-GES): Standart GES sunucunun sağladığı bütün fonksiyonları sağlamanın yanısıra, GES sunucuların kimlik denetimi, politika ve anahtar yönetimi işlerini yürütür. Sistemde yer alacak her GES sunucu öncelikle kimlik denetiminden geçmek zorundadır. Güvenlik Denetleyici düzende çalışan GES sunucu her GES sunucu için bir IP-anahtar ikilisini kimlik denetimi için veri tabanında tutar. Kimlik denetiminden geçmek isteyen GES aktif hale gelirken ilgili anahtarı GD-GES'e iletir. GD-GES, kendisine ulaşan anahtar ile istekte bulunan GES sunucunun IP adresini veri tabanındakiler ile karşılaştırır, uygun bir kaydın bulunması durumunda GES kimlik denetiminden geçer. Kimlik denetiminden geçen GES, GD-GES ten diğer GES sunucular ile haberleşeceği zaman kullanacağı bileti alır. Diğer GES sunucular kendisi ile haberleşmek isteyen GES sunucunun biletini GD-GES' ten doğruladıktan sonra GES sunucular arası iletişim başlar. Her GES sunucu, çalıştığı düzenden bağımsız olarak, sistemde yer almadan önce özel-açık anahtar ikilisini yaratır ve açık anahtarı GD-GES'e iletir. GD-GES sistemdeki bütün GES sunucuların açık anahtarını veri tabanında tutar ve diğer sunuculara iletir. Böylece merkezi olarak açık anahtarların dağıtımı gerçeklenir. Diğer GES sunucuların açık anahtarlarına sahip olan her GES artık SSL ile haberleşmeye hazır durumdadır.

GD-GES'in önemli bir elemanı politika yaratıcısıdır. Politika yaratıcısı farklı politikalar yaratabilir ve farklı GES sunucular üzerine bu politikaları yükleyebilir.

Disk erişimi ve soket bağlantısı oluşturarak ağ kaynaklarını kullanma gibi etmen aktiviteleri, yaratılan yeni bir Java *Security Manager* ile izlenir ve GES üzerinde yüklü olan politikaya bağlı olarak bu aktiviteye izin verilir veya verilmez. GES sunucu aynı zamanda etmen haberleşmesi ve etmen göçleri gibi aktiviteleri de güvenlik politikaları uygulayarak sınırlayabilir. Herhangi bir anda belirli bir GES üzerinde sadece bir politika aktif olabilir; ancak politika yaratıcısı istenilen zamanda yeni bir politikayı GES üzerine yükleyebilir.

3.2 GÜVENLİK POLİTİKALARI

Güvenli Etmen Sistemi bütün etmen aktivitelerinin güvenlik politikalarının izin verdiği ölçüde gerçeklendiğini, politika kurallarını işleyerek garanti eder. Haberleşme, göç, diske yazma veya diskten okuma, sistem kaynaklarına erişim gibi aktiviteler güvenlik politikaları ile kontrol edilen aktivitelerin bazılarıdır. Güvenlik politikaları GD-GES tarafından (Politika Yaratıcısı) yaratılır ve diğer GES sunuculara iletilir. Politikalar aşağıda belirtilen amaçları gerçeklemek için kullanılabilir:

- Bir etmenin sadece belirlenen etmenler ile haberleşmesine izin verilebilir; etmenin sadece belirlenen düğümlerdeki etmenlere mesaj göndermesi veya bu düğümlerden mesaj alması sağlanabilir. Etmenin mesaj göndermesi veya alması ayrı ayrı kısıtlanabilir. Zaman sınırı verilerek etmenin belirli bir zaman aralığında mesaj göndermesi veya alması da sağlanabilir.
- Bir etmenin sadece belirli düğümlere göç etmesine izin verilebilir. Bir düğümün sadece belirli düğümlerden etmen kabul etmesi sağlanabilir. Zaman tabanlı kısıtlamalar da mümkün olabilir.
- Etmenin düğüm üzerinde hiç bir disk aktivitesinde bulunmaması istenebilir. Sadece belirli etmenlerin diske yazması veya diskten okuması sağlanabilir. Zaman sınırı vermek mümkündür.
- Etmenin soket nesnesi yaratarak ağ kaynaklarını kullanması engellenebilir. Etmenin bağlantı kurma veya bağlantı isteği kabul etme istemleri ayrı ayrı sınırlandırılabilir. Zaman sınırı vermek mümkündür.
- Etmenin sistem değişkenlerine erişimi tümden veya zamana bağımlı olarak sınırlandırılabilir.

Şekil 3, GES sunucularının uyguladığı bir politikaya ait kuralları göstermektedir. Örneğin Kural 1, "Agent1" isimli bir etmenin bulunduğu yerden bağımsız olarak "Agent2" isimli etmene mesaj gönderebileceğini söylemektedir. Kural 2 , "Agent1" isimli etmenin 172.1.1.1 IP adresli düğüm üzerinde çalışan "AgentServer" isimli GES sunucu üzerindeki hiç bir etmene mesaj gönderemeyeceğini söylemektedir. "Agent4" etmeninin kural 6 ve kural 7 ile 172.1.1.1 düğümü izerinde disk erişimi okuma ile sınırlandırılmıştır.

Rule No	Source Agent	Target Agent	Destination Server	Service	Time	Action
1	Agent1	Agent2	Any	SendMessage	Any	Permit
2	Agent1	Any	172.1.1.1/AgentServer	SendMessage	Any	Block
3	Any	Agent2	172.1.1.1/AgentServer	ReceiveMessage	Any	Permit
4	Agent3	Null	172.1.1.2/AgentServer	Move	Any	Block
5	Any	Null	172.1.1.1/AgentServer	Move	01.01.2004 13:00 01.01.2004 14:00	Permit
6	Agent4	Null	172.1.1.1/AgentServer	ReadDisk	Any	Permit
7	Agent4	Null	172.1.1.1/AgentServer	WriteDisk	Any	Block
8	Any	Null	172.1.1.1/AgentServer	AcceptConnection	Any	Block
9	Any	Null	172.1.1.1/AgentServer	RequestConnection	Any	Permit
10	Any	Any	Any	Any	Any	Block

Şekil3. Örnek bir politika kural dizisi

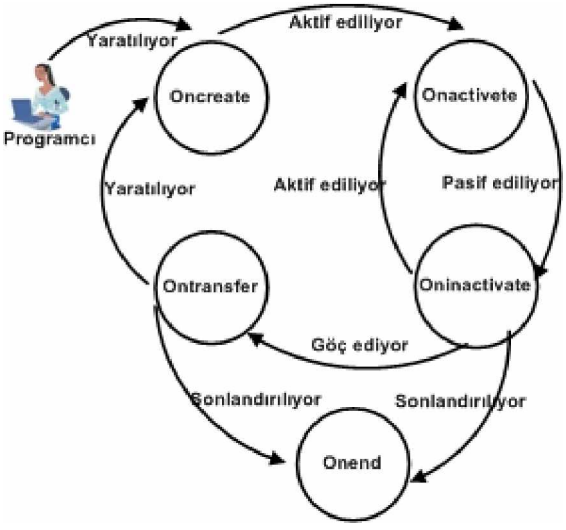
Politika kuralları sırasıyla kontrol edilir ve eğer aktiviteye uygun bir kural bulunursa GES sunucu kuralın belirlediği sınırlandırmayı uygular ve aktiviteye ait kural taraması sonlandırılır.

Kuraldaki hedef sunucu (DestinationServer) alanı iki bölümden oluşmaktadır. İlk bölüm GES sunucunun üzerinde çalıştığı düğümün IP adresini vermektedir. İkinci bölüm ise GES sunucu adını belirtir. Teorik olarak aynı düğüm üzerinde farklı isimlerle birden çok GES sunucu çalışabilir ancak bu çalışma şekli denenmemiştir.

Yukarıda açıklanan kurallar şeklinde tanımlanmayan ancak güvenlik için ayarlanması gereken başka parametreler de mevcuttur. Örneğin, bir etmenin belirli bir sınırın üzerinde bellek alanı kullanması engellenebilir. GES sunucu etmene ait nesne ağacını tarar ve etmen'e ait bütün nesnelerin işgal ettiği bellek alanını hesaplar. Verilen sınırı aşan etmenler pasif duruma geçirilir. Diğer bir parametre, isimsiz etmenlerin çalışmasına izin verilmemesidir. Bir etmen kendini bir isimle duyurup çevresine karşı görünür duruma gelir. Diğer etmenler bu etmenle ismini sorgulayarak etkileşime geçebilir. Güvenlik politikaları etmen ismine göre işletildiği için kendini duyurmayarak görünür duruma gelmemiş bir etmenin çalışması sakıncalı görülebilir. Mimari, aynı isimle birden çok etmenin kendini aynı GES sunucu üzerinde duyurmasına izin vermez, ancak aynı isimli birden çok etmen farklı GES sunucular üzerinde çalışıyor olabilir.

4. GES ETMENLERİ

Bir GES etmeni, yaşam döngüsü içinde, *yaratılma*, *aktif olma*, *iletilme*, *pasif olma* ve *yok edilme* temel durumlarınının birinde olur. Şekil 4, bu durumlar arasındaki geçişleri göstermektedir.



Şekil 4. Bir etmenin yaşam döngüsü.

Yaratılma durumu: Etmenin, ağdaki bir düğüm üzerinde yaratılmaya başlanmasına iki farklı nedenden dolayı başlanabilir. Bunlardan ilki etmeni yazan programcının etmene ait kodu düğüm üzerindeki GES sunucuya iletmesi, ikincisi ise etmenin ağdaki diğer bir düğüm üzerinden bu düğüm üzerine taşınmasıdır. İkinci durumda kaynak GES sunucu, iletim başarılı ise etmeni üzerinden siler. Yaratılma anında *AgentIdentity* sınıfından bir kimlik bilgisi oluşturulur ve etmene atanır. Sistemdeki bütün etmenlere erişim kimlik bilgisi üzerinden yapılmaktadır. Etmen kimlik bilgisi 3 parçadan oluşur. İlk parça (ID) 128 sekizli gelişigüzel oluşturulan bir katardır ve etmen yaşam süresi boyunca değişmez. İkinci parça (strVisibleAgentName) etmenin kendini duyurduğu ve diğer etmenlerin etmen kimliğini öğrenmek için kullandığı ismidir. Üçüncü parça (strAgentHostName) ise etmenin o an üzerinde çalıştığı GES sunucu adresidir ve etmen göç ettiğinde değişir. GES sunucu adresleri IP adresi-isim ikilisinden oluşmaktadır. IP adresi GES sunucunun üzerinde çalışığı düğümün IP adresini, isim ise GES sunucu adını gösterir.

```
public final class AgentIdentity
    implements serializable {
    private String strAgentHostName=null;
    private String strVisibleAgentName=null;
    private byte [] ID =new byte[128] };
```

Bir etmen haberleşeceği etmenin kimlik bilgisini bilmek zorundadır. Etmene verilen arayüz (*AgentInterface*) etmen isminden kimlik bilgisinin öğrenilebileceği esnek fonksiyonlar içermektedir. Etmen yaratıldığında, gerekli veri yapıları oluşturulur, etmen kimlik bilgisi atanır.

Aktif durumu: Bu durumda etmen aktif olur ve çalışmaya başlar. Etmen diğer etmenler ile haberleşmeye başlaması için aktif durumda olmalıdır. Programcı bu metoda ilişkin herhangi bir koda parçası yazmak zorunda değildir. Aktif duruma geçip *OnMessageArrive* metoduna yazılan kod ile gelen mesajları işleyebilir.

Pasif durumu: Etmen pasif duruma düştüğünde çalışması durdurulur ve etmene ait kod ve durum bilgisi GES sunucu tarafından diske yazılır. Etmen bu durumda iken çevresi ile etkileşim içinde olamaz.

İletilme durumu: Aktif olan bir etmen herhangi bir anda başka bir düğüm üzerine taşınma isteğinde bulunabilir. GES sunucu transfer işlemi başlamadan önce etmeni pasif duruma getirir. Daha sonra etmen kodunu ve verisini karşı tarafa iletir ve iletim başarılı ise etmene ait kod ve durum bilgisi diskten silinir. Bu arada uzak GES sunucu etmeni karşı tarafta yeniden yaratır, etmene ait son durum bilgisini yükler ve etmeni aktif duruma getirir. GES mimarisi güçlü göçü desteklemez, bu nedenle hedef düğüm üzerinde etmene ait veriler günceldir ancak etmen kaldığı satırdan çalışmaya devam etmek yerine aktif duruma ait kodun

ilk satırından çalışmasına devam eder.
Sonlanma durumu: Etmen, başarılı iletim sonucu yerel disk üzerinden silinir.
GES mimarisi, etmeni geliştirecek programcıya esnek bir ortam sunacak şekilde tasarlanmıştır. Programcı aşağıda gösterilen etmen şablonuna bağlı kalarak etmenin davranışlarını programlar. Etmen haberleşmesi ve hareketliliği için sistem tarafından sunulan fonksiyonları kullanması yeterli olur.

```
public class Main extends Agent{
    public void OnMessageArrive(){... }
    public void OnCreate(){ ... }
    public void OnActivate(){... }
    public void OnInactivate(){... }
    public void OnTransfer(){... }
    public void OnEnd(){... }}
```

Programcının etmenin her durumuna ilişkin kodu ilgili metod içinde yazması beklenir. Örneğin *OnCreate()* metodu içinde etmen yaratılırken yapılması istenilenler, *OnMessageArrive()* metodu içinde etmene mesaj geldiğinde yapılması istenenler yazılmalıdır.

4. 1. ETMEN HABERLEŞMESİ

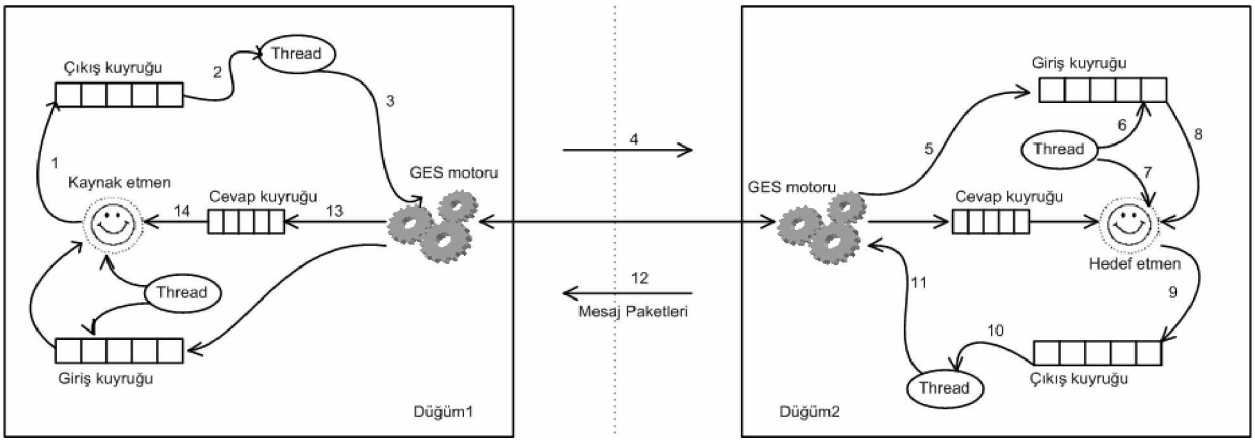
GES etmenleri birbirleri ile mesajlar aracılığı ile haberleşirler. Mimari *AgentInterface* nesnesinin sağladığı metodlar ile asenkron haberleşmeye destek verir. Bütün etmen haberleşmeleri SSL ile şifreli olarak gerçeklenir. Etmenlere, mesaj iletebilecekleri, mesaj akıbetlerini sorgulayabilecekleri, mesaj yanıtı için belirli bir süre bekleyebilecekleri esnek fonksiyonlar sunulmaktadır.

Şekil 5. etmen tarafından gönderilen bir mesajın iletim aşamalarını göstermektedir. Her yeni etmenin yaratılma aşamasında, etmeni sarmalayan kabuk nesne içinde etmene ait bir mesaj giriş kuyruğu, bir mesaj çıkış kuyruğu ve bir de yanıt kuyruğu oluşturulur. Etmen aktif duruma geçtiğinde, giriş ve çıkış kuyruklarını sürekli gözleyen birer *iplik* canlandırılır. Giriş kuyruğunu izleyen iplik, yeni bir mesaj ulaştığında etmeni durumdan haberdar eder. Çıkış kuyruğunu izleyen iplik ise, yeni bir mesajın kuyruğa eklenmesi üzerine, GES motorunu uyarır.

Sistemdeki bir alıcı etmene mesaj iletmek isteyen gönderici etmen, bir çağrı ile bu isteğini bildirir. Çağrı istek mesajını, mesaj çıkış kuyruğuna yerleştirir. Bu noktadan sonra, etmen iletimin gerçeklenme ayrıntılarıyla ilgilenmez. Eğer bir yanıt bekliyorsa, mesaj yanıtı gelene kadar diğer işleriyle ilgilenebilir. Dilerse, haberleşme isteğinin sonucunu da sorgulayabilir. Bu arada, çıkış kuyruğunu gözleyen iplik, mesaj iletim isteğini GES motoruna aktarır. GES motoru alıcı etmenin üzerinde yer aldığı düğümde etkin olan GES motoruyla yaptığı işbirliği sonucu mesajın alıcı etmenin giriş kuyruğuna yerleştirilmesini sağlar. Giriş kuyruğunu gözleyen iplik, derhal alıcı etmeni uyarır. Etmen bu uyarıdan sonra, uygun gördüğü anda mesajı alma isteğinde bulunabilir. Yanıtlar da mesaj şeklinde iletilir, ancak gönderici kabuk, yanıt mesajlarını paketin sonuna özel bir işaret (ACK) koyarak, GES motorunun yanıt mesajını karşı tarafın yanıt kuyruğuna yerleştirmesini sağlar. Gönderici etmen yanıt kuyruğundan istediği zaman yanıtları çekebilir.

"`Message`" sınıfından bir nesne ile tanımlanan her mesajın bir ismi ve parametre listesi mevcuttur ve "`sendMessage`" çağrısı ile mesajı iletme isteğinde bulunulur. Bu çağrının geri dönüş değeri ile mesaj iletiminin durumu hakkında bilgi edinmek mümkündür. Örneğin ismi "Topla" olan ve iki sayının toplamını öğrenmek için gönderilecek bir mesaj aşağıdaki şekilde oluşturulur.

```
Message message = newMessage
                        ("Topla","123","456");
```



Şekil 5. Etmen haberleşme alt yapısı

Parametreler, "*serializable*" olma özelliğini taşıdığı takdirde herhangi bir tipten olabilir. İkinci adım, yerel GES sunucu ile haberleşmek için arayüz isteğinde bulunmaktır.

```
AgentInterface
        agentinterface=getAgentInterface();
```

Bir sonraki adım, kaynak etmenin, ismini bildiği hedef etmenin kimliğini öğrenme aşamasıdır.

```
AgentIdentity targetagentidentity =
Agentinterface.getVisibleAgentIdentity
                              ("Hesaplayici");
```

Son olarak mesaj "`sendMessage`" çağrısı ile gönderilir. "`sendMessage`" çağrısının geri dönüş değeri "`Pidendifier`" sınıfından bir nesnedir ve etmenin mesaj akıbetini öğrenmesi için veya mesaj yanıtını alması için kullanılır.

```
Pidentifier id=agentinterface.sendMessage
       (targetagentidentity,message);
if (!agentinterface.waitForReply(id,5000))
//yanıt için 5 sn. bekle
   System.out.println("Timeout.."+"\n");
Else {String result =
  (String)agentinterface.getReply(id);
 System.out.println("123+456"+"="+result);}
```

Alıcı etmen mesajı gelen paket içinden alır, parametreleri çözer ve işlemi yapıp ilgili paketin yanıtını gönderir. Alıcı etmen bu işlemleri ister "`OnMessageArrive`" metodu içinde isterse "`OnActivate`" metodu içerisinde gerçekleyebilir. Ancak "`OnActivate`" metodunda gerçekliyor ise, yeni bir mesajın varlığını "`waitForMessage`" çağrısı ile kendi kontrol etmelidir. Aşağıdaki kod parçası alıcı etmenin "`OnMessageArrive`" metoduna aittir.

```
Packet
      packet=getAgentInterface().receive();
Message message =
              (Message) packet.getObject();
Object[] parameters =
               message.getParameters();
String par1 = (String) parameters[0];
String par2 = (String) parameters[1];
int returnValue =
       Integer.valueOf(par1).intValue() +
       Integer.valueOf(par2).intValue();
getAgentInterface().sendReply(packet,
               String.valueOf(result));
```

4.2. ETMEN GÖÇLERİ

GES mimarisi zayıf etmen göçünü destekler. Etmen AgentInterface nesnesinin Move çağrısı ile göç etme isteğini bildirir. Etmen göç etmek istediği uzak GES sunucu adresini vermek zorundadır. Göç etme sırasında GES sunucular aşağıdaki adımları yürütürler:

- Etmen yerel GES sunucu üzerinde pasif yapılır ve etmene ait pasif durumu bir mesaj ile Gözleyici düzende çalışan GES sunucuya iletilir.
- Etmen kod ve durum bilgisi yerel GES sunucu diskine yazılır. Etmenler Java dili ile yazıldıklarından birden çok sınıf dosyasından oluşabilir. Etmene ait bütün sınıf dosyaları bir sıkıştırılmış dosya içinde tutulur, böylece hem iletim karmaşıklığı hem de iletim süresi azaltılmış olur. Etmen kodu ve verisi diske yazılırken DES şifreleme algoritması ile şifrelenir. Bunları çözecek doğru anahtar ise sadece Güvenlik Denetleyicisi düzeninde çalışan GES sunucudan öğrenilebilir.
- Etmen kod ve durumunu içeren dosyalar uzak GES sunucuya iletilir.
- Uzak GES sunucu, etmene ait kod ve durum bilgisini içeren şifreli dosyaların şifrelerini çözerek etmeni yeniden yaratır, kimlik bilgisini günceller, son durumunu yükler ve etmeni aktif duruma geçirir. Etmene ait aktif durum bilgisini içeren bir mesaj Gözleyici düzende çalışan GES sunucuya iletilerek etmene ait yer bilgisi güncellenir.
- Bütün bu aşamalar başarılı ise, etmen kaynak GES sunucu üzerinden silinir.

Bu aşamalardan herhangi biri başarısız olursa GES sunucular transfer işlemini sonlandırır. Etmen, göç etme isteğini Move(adres) çağrısı ile bildirdiğinde, geri dönüş değeri olarak çağrı sonucunu öğrenebileceği TransferResult sınıfından bir nesne döndürülür. Çağrıdaki adres parametresi hedef düğümün IP adresini ve bu düğüm üzerindeki GES sunucunun adını içermektedir. Aşağıdaki kod parçası, başarılı bir göç etme gerçeklenene kadar göç etme isteğini farklı bir düğüm için tekrarlayan bir etmene aittir.

```
  while (jobs_todo) do_jobs();
  transferresult=false;
  while((address!=null)||(!transferresult))
  {TransferResponse transferresult =
   getAgentInterface().Move(address);
  if(transferresult.FAILED)
       address=newaddress();
else transferresult=true;}
```

Etmen göç etme işlemi başarılı şekilde sonlandığı zaman etmene ait yer bilgisi otomatik olarak Gözleyici düzende çalışan GES sunucu üzerinde güncellenir; böylece etmene gönderilecek yeni mesajlar doğru GES üzerine yönlendirilebilecektir. Etmen iletim sırasında mesaj gönderemez veya alamaz. Bu yüzden bir diğer etmene mesaj yollayan etmen, mesaj sonucunu sorgulamalı ve gerekiyorsa mesajı yeniden yollamalıdır. GES sunucular

arasında iletim sırasında gerçekleşen bütün haberleşmeler SSL ile şifrelenir.

5. SONUÇ

Bu bildiride güvenli bir hareketli etmen mimarisi olan GES'in tasarım ve gerçekleme ayrıntılarına değinilmiştir. GES, sarmalanmış etmen modeli ile etmeni çevresinden korur. Güvenli etmen haberleşmesi ve göçü için gerekli fonksiyonları sunar. Yetkisiz etmen aktiviteleri için güvenlik politikalarının kullanımını destekler. Sistem özellikle sadece etmenlerin değil, düğümlerin de karşı karşıya olduğu güvenlik tehlikelerine karşı geliştirilmiştir. Güvenlik özellikleri mimariye tasarım aşamasında yerleştiriliştir. Mimari esnek ve geliştirilebilir bir yapıya sahiptir.

Sadece GES tarafından değil diğer hareketli etmen mimarileri tarafından da engellenemeyecek atak türleri bulunmaktadır. Örneğin bir etmenin sonsuz döngüye girerek düğüm üzerinde aşırı işlemci yükü oluşturmasını engellemek için belirgin bir yöntem bulunmamaktadır. Etmenlerin her biri iplik olarak çalıştıkları için iplik önceliğini değiştirmek bu türden atakları engellemez ancak düğüme ait kaynakların daha iyi kullanımını sağlayabilir. Çalışmanın ilerleyen aşamalarında güvenlik politikalarına etmen öncelik özelliğinin de eklenmesi planlanmaktadır. Aşırı bellek kullanımı yüzünden oluşabilecek hizmet kıtlığı atakları GES tarafından engellenebilir. GES sunucuları, kök nesne etmen olmak üzere, nesne ağacını tarar ve nesnelerin kullandığı toplam bellek alanını hesaplayıp belirli bir değeri aşması halinde etmenin çalışmasını sonlandırabilir. Etmen aktivitelerinin izleri GES sunucuları tarafından kaydedildiği için, ilk aşamada engellenemeyen hizmet kıtlığı atakları sonraki aşamalarda engellenebilir. İzlerin analiz edilip dinamik kurallar ile güvenlik politikalarının çalışma anında otomatik olarak güncellenmesi mimariye eklemlenmesi düşünülen diğer bir özelliktir. İkincil Güvenlik Denetleyici ve Gözleyici GES sunucularının da desteklenmesi planlanan çalışmalar arasındadır. Güvenlik politikalarının detaylandırılması, örneğin diskten okuma ve yazma gibi genel yetkilendirmeler dışında belirli dosyalardan okuma veya belirli dosyalara yazma gibi yetkilendirmelerin verilebilmesi de mümkün olacaktır.

6. KAYNAKLAR

[1] Gopalan Suresh Raj "A Detailed Comparison of CORBA, DCOM and Java/RMI" http://my.execpc.com/~gopalan/misc/ compare.html

[2] S. Franklin and A. Graesser "Is it an Agent, or just a program? A taxonomy for Autonous Agents" Proc. Third International Workshop on Agent Theories, Architecures,and Languages, Springer -Verlag,1996.

[3] Karnik, N.M., Tripathi, A.R., 1998. Design issues in mobile-agent programming systems. IEEE Concurrency 6 (3), 52–61

[4] Telescript, http://www.science.gmu.edu/ ~mchacko/Telescript/docs/telescript.html

[5] Tacoma, http://www.cs.uit.no/forskning/DOS/Tacoma/

[6] AgentTcl, http://agent.cs.dartmouth.edu/ general/ agenttcl.html

[7] Aglets, http://researchweb.watson.ibm.com/trl /projects/ aglets/index_e.htm

[8] Voyager, http://www.recursionsw.com/ products/ voyager/voyager.asp

[9] Concordia, http://www.merl.com/projects/concordia/

[10] Ajanta, http://www.cs.umn.edu/Ajanta

[11] Tschudin, Christian: Mobile Agent Security. In: Matthias Klusch (Ed.): Intelligent information agents: agent based information discovery and management in the Internet, pp. 431 - 446, Springer-Verlag. 1999

[12] V.Varadharajan and D.Foster ,"A Security Architecture for Mobile Agent Based Applications" World Wide Web:Internet and Web Information System, 6,93-122, Kluwer Academic Publishers 2003

[13] F.B. Schneider, "Towards Fault-Tolerant and Secure Agentry,"Proceedings 11th International Workshop on Distributed Algorithms, Saarbucken, Germany,September 1997

[14] Young and M.Yung," Sliding encryption, A Cryptographic Tool for Mobile Agents" Proc. 4th Int. Wksp. Fast Software Encryption , Springer-Verlag 1997

[15] F. Hohl, "Protecting mobile agents with blackbox security" Proc. 1997 Wksp. Mobile Agents and Security , Univ. of Maryland , Oct 1997

[16] T. Sander, "On cryptographic protection of mobile agents" Proc. 1997 Wksp. Mobile Agents and Security , Oct 1997

[17] Uwe G. Wilhelm and Sebastian Staaman "Protecting the itinerary of Mobile Agents" Laboratoire de Systemes d'Exploitation, Switzerland,1998

[18] Vipin Swarup "Trust Appraisal and Secure Routing of Mobile Agents" The Mitre Corporation 1997

[19] Wayne Jansen,Tom Karygiannis "Mobile Agent Security" NIST Special Publication 800-19

[20] D. Gollman, Computer Security, John Wiley & Sons: New York, 1999