

Dağıtılmış Sistem Programlamada İlgiye Yönelik Yaklaşım ile Güvenli Mesajlaşma

Özgür Koray Şahingöz¹

¹ Hava Harp Okulu Komutanlığı, Bilgisayar Mühendisliği Bölümü, 34149, Yeşilyurt-İSTANBUL

o.sahingoz@hho.edu.tr

Özet: Dağıtılmış Sistem programlaması başlangıçta dağıtılmış nesnelere gerçekleştirilmekteyken, gelişen teknoloji ile birlikte günümüzde aktif nesnelere, aktörler, etmenler ve benzeri programlama paradigmaları da etkin şekilde kullanılmaktadır. Bu gibi sistemlerin yapısı gereği haberleşme güvenli olmayan bir arabirim olan İnternet üzerinden yapılmaktadır. Bu yazılım mimarilerinde güvenli haberleşmenin sağlanması ilgi çekici olmakla birlikte gerçekleştirilmesi yazılım mühendisliği açısından zor bir konudur. Yazılım mühendisliğinin yapısı gereği programcılar belirli konularda uzmanlaşmaktadır. Dağıtılmış Sistem programlaması konusunda uzmanlaşan bir kişinin, aynı zamanda güvenlik metodolojileri konusunda da uzman olması beklenemez. Onun için bu iki olguyu (ilgiyi) bir birinden ayıracak şekilde bir yazılım mimarisi geliştirmek önemlidir. Ama dağıtılmış programlamanın yapısı gereği, bileşenler arasındaki mesajlaşma işlevi sistemdeki çok farklı bileşenler üzerine dağıtılmış ve bir birleri ile kesişen ilgiler oluşturmaktadır. Dağıtılmış sistemlerdeki bu problemin çözümünde İlgiye Yönelik Programlamanın¹ yeni bir yaklaşım getireceği ve mimari geliştirimini kolaylaştıracağı öngörülmektedir. Bu bildiride daha önceden geliştirilmiş olan bir dağıtılmış sistemin, ilgiye yönelik yaklaşımla nasıl güvenli haberleştirilebileceği anlatılmış ve ilgiye yönelik programlamanın avantajları sıralanmıştır.

Anahtar Sözcükler: İlgiye Yönelik Programlama, Dağıtılmış Programlama, Güvenli Mesajlaşma

Secure Communication in Distributed System Programming with Aspect Oriented Approach

Abstract: The programming of distributed system was firstly implemented with objects. However, because of technological advances, active objects, actors, agents and other types of programming paradigms are used nowadays. These types of systems are generally using İnternet, which is an insecure communication platform, for transferring data. It is a difficult task to make a secure communication between distributed objects on different machines. Due to the nature of software engineering, programmers cannot be specialized all areas of system development. While some programmers specialized on distributed system programming, others specialized on database management and some others specialized on security management. Therefore, it is important to disseminate these two important concerns, distributed programming, and security programming, from each other. Because the usage of distributed objects, some concerns can be scattered on different objects, and this results some crosscutting concerns on the system. Aspect Oriented Programming concept brings a new

¹ Aspect Oriented Programming

paradigm to solve this type of problems and it is expected to ease the development of distributed system programming. In this paper, how one can import aspect oriented features to a pre-developed distributed system for enabling secure object communication is described and advantages of aspect oriented programming is listed

Keywords: Aspect Oriented Programming, Distributed Programming, Secure Messaging

1. Giriş

Günümüzdeki büyük yazılım endüstrileri kapsamlı bir yazılım mimarisi geliştirirken gerek ülkenin gerekse dünyanın değişik bölgelerine coğrafi olarak yayılmış yazılım geliştirme takımları kullanmaktadır. Bu şekilde büyük boyutlu yazılım geliştirilmesinde yazılımların farklı özelliklerini gerçeklerken, her takımın ortak güvenlik mekanizması geliştirmesi, ortak protokolleri kullanmaları gibi güvenlik konularında tekliğin sağlanmasında zorlanılmakta ve yazılımın uzun süreli olarak idamesi bu endüstri kolunda ciddi bir sorun olarak karşımıza çıkmaktadır.

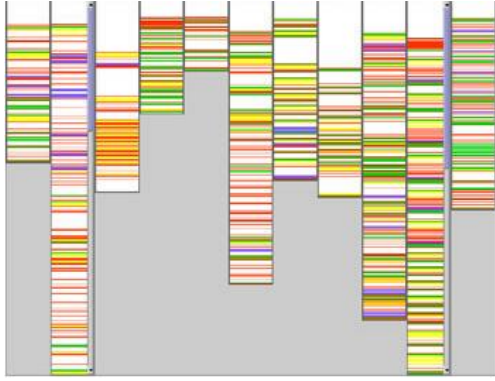
Yazılım takımlarını ortak hareket etmelerinin haricinde yazılım geliştiricilerin güvenlik konusunda istenilen uzmanlıkta olmaları genellikle pek mümkün değildir. Güvenlik kendi içerisinde özel bir konu ayrı bir çalışma alanı ve uzmanlık gerektirmektedir. Bu nedenle güvenlik kısmının ayrı bir modül olarak geliştirilmesi önem arz etmektedir. Ancak yapısı gereği güvenlik mekanizması dağıtılmış yazılım mimarileri içerisinde bir *kesişen ilgi*² durumunda olup aynı birden çok nesne içerisinde farklı farklı yerlerde gerçekleşmektedir. Bunun sonucu olarakta bu modülerliği sağlamak için kesişen ilgilerle uğraşan bir yazılım metodolojisi olan İlgiye Yönelik Programlama yaklaşımından faydalanılması hem yazılımın ilk geliştirilmesinde hemde idamesinde büyük kolaylıklar sağlayacaktır.

Bazı tahminler %50-90 arasındaki yazılım geliştirme kaynaklarının yazılım idamesi için ayrıldığını göstermektedir. Yazılımı uzun süreli idame ettirebilmek için iyi bir dokümantasyonun yanı sıra sistemin sağlam bir modülerlikte geliştirilmiş olması önemlidir. Nesneye Yönelik Programlama (NYP) bu alanda çok önemli faydalar sağlasa da, kesişen ilgiler gibi bazı noktalarda modülerlik konusunda eksik kalmaktadır.

NYP yaklaşımında her bir işlev bir sınıf üzerine verilmekte ve o sınıf içerisinde uygun bir metod çağırımı ile halledilmektedir. Ancak, güvenlik, log alma, veri tabanı erişimi, hata yakalama vb. gibi bazı işlevler sadece tek bir sınıfa yüklenememektedir, farklı sınıflar içerisinde bu işlevin (ilginin) enine kesen (crosscutting) bir yaklaşımla ayrı ayrı yeniden kodlanmasını gerektirmektedir. Yaklaşık 20 bin satırlık kod içeren nispeten basit bir yazılım sistemindeki enine kesen işlevler Şekil 1'de farklı renklerde gösterilmiştir.

Enine kesen ilgi yaklaşımı iki temel problemi beraberinde getirmektedir: Öncelikle, bazı işlevlerin bu şekilde sistemdeki birçok sınıfını içerisine *dağıtılması* (scattering) modülerliği engellemektedir. Aynı zamanda, modülerliği azaltan diğer yaklaşımda bir sınıf içerisine birden fazla işlevin yüklenmesiyle de bir *karmaşıklık* (tangling) olarak karşımıza çıkmaktadır.

² crosscutting concern



Şekil 1. Çok Sınıflı Bir Yazılım Mimarisinde Enine Kesen İşlevler.

İlgiye Yönelik Programlama (İYP) yaklaşımı nispeten yeni ortaya çıkan bir metodoloji olup, enine kesen işlevleri birbirlerinden daha iyi bir şekilde ayırarak, yazılımın modülerliğini ve kalitesini artırmayı amaçlamaktadır [1][2]. Bu yaklaşımın dağıtılmış bir yazılım mimarisi geliştirmede daha etkin ve verimli olacağı değerlendirilmektedir.

Güvenli haberleşmenin yapılması temelde iki seçenekle gerçekleştirilebilir. Birinci yaklaşımda her bir (aktif) nesne kendi güvenlik mekanizmasını gerçekleştirir. Bu durum yazılım mimarisinden tamamen bağımsız olacağı için bir tekillik arz etmesi zordur. İkinci yaklaşım ise yazılım mimarisinin çalıştığı platformun güvenli olarak kabul edilmesidir. Bu da sisteme dahil olacak her bir (aktif) nesnenin güvenlik kontrollerinin yapılması ile sağlanabilir. Ancak günümüzde açık sistem yapısı sayesinde bu güvenlik mekanizmasının platform tarafından sağlanmasında ciddi zorluklarla karşılaşmaktadır. Bu nedenle her uygulamanın kendi gereksinimini karşılayacağı ve güvenlik politikalarında tekilliğin ve modülerliğin sağlanacağı bir yapının kurulması önemlidir.

Bu bildiride önceden geliştirilmiş olan mesaj haberleşmeli bir dağıtılmış programlama mimarisinde haberleşme güvenliğinin *İlgiye Yönelik Yaklaşım*la nasıl sağlanabileceği gösterilmekte ve önerilen yazılım mimarisinin avantajları detaylandırılmaktadır. Bildirinin ilerleyen bölümleri şu şekilde yapılandırılmıştır: 2. Bölümde Dağıtılmış Sistemlerde Haberleşme Güvenliği konusu irdelenmiş, 3. ve 4. Bölümlerde sırasıyla İYP yaklaşımı ve AspectJ yazılımı incelenmiştir. Önerilen yazılım mimarisinin detayları 5. Bölümde gösterilmiş olup son bölümde Sonuç ve Gelecek Çalışmalar vurgulanmıştır.

2. Dağıtılmış Sistemlerde Haberleşme Güvenliği

Dağıtılmış sistemlerde haberleşme genellikle Internet gibi güvenli olmayan ortamlar üzerinden yapılmaktadır. Bu hatlarda verinin izlenmesi ve içeriğinin ele geçirilme olasılığı yüksektir. Bu nedenle kritik yazılımlarda transfer edilen verinin güvenli olarak karşıya ulaştırılmasında şifreleme yapılması kaçınılmazdır. Dağıtılmış yazılım mimarilerinin programlamasında gerek asıllama işlemleri için gerekse güvenli veri transferi işlemleri için farklı protokollerin kullanılması gerekebilir. Güvenlik yazılımcıları daha önce geliştirilmiş olan protokolleri kullanılabilecekleri gibi kendi özel protokollerini de geliştirebilir ve verileri şifreli olarak alıcıya/alıcılara iletebilirler.

Şifreleme işlemlerinde iki tip şifreleme kullanılır: simetrik şifreleme ve asimetrik şifreleme. Şifreleme ve şifre çözme işlemlerinin birbirlerinden farklı anahtarlarla yapıldığı asimetrik şifreleme işlemleri daha güvenilir olmasına rağmen daha fazla işlemci gücü ve işlem zamanı gerektirdiğinden büyük boyutlu verilerin transferi işlemlerinde veya gerçek zamanlı veri aktarılması işlemlerinde fazla tercih edilmezler. Şifreleme ve şifre çözme işlemlerinde ortak anahtarların kullanıldığı simetrik şifreleme de

daha hızlı çalışmasına rağmen aynı şifre kullanıldığı için yeterince güvenli değildir. Bunun yerine asimetrik anahtarlar kullanılarak simetrik bir oturum şifrenin üretilmesi daha sonra belirli süre için üretilen bu oturum şifrenin kullanılması ise birçok yazılım geliştirici tarafından tercih edilmiştir. Oturum sonunda ise simetrik anahtar değiştirilir veya kullanılmayarak bir sonraki oturumun kurulması sırasında yeniden oluşturulur.

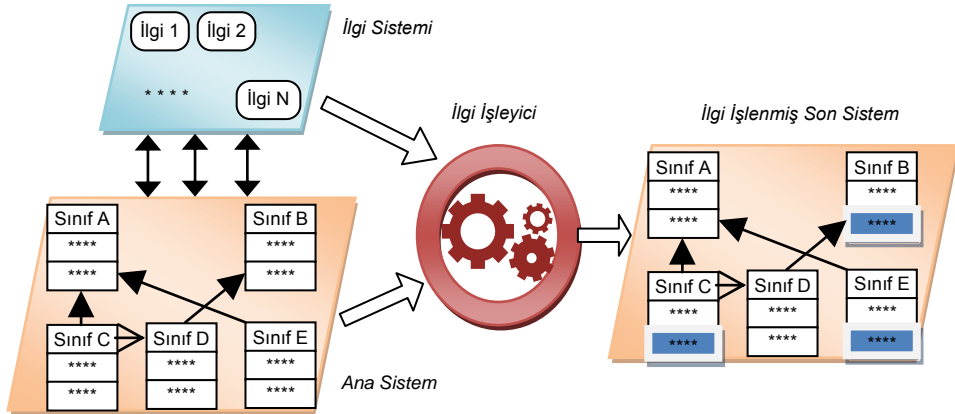
Mesaj transferi özellikle paralel ve/veya dağıtılmış olarak çalışan mimarilerde kullanılan bir haberleşme paradigmasıdır. Mesaj gönderilmesi işlemi uzaktan metod çağırılması ile veya soketler üzerinde veri paketleri gönderilmesi ile sağlanabilir. Bu işlemleri yaparken bazı güvenlik gereksinimlerini göz önünde bulundurmak gerekmektedir. Böyle mimarilerde tek bir güvenlik modeli her uygulama modeli için ihtiyaç duyulan farklı kriterleri sağlamayabilmektedir. Haberleşme güvenliğini sağlarken gerçek zamanlı veri iletimi ile güvenilir veri iletimi arasındaki dengenin uygulama katmanında kullanılan mimari modele göre sağlanabilmesi önemlidir. Örneğin şifreleme işlemlerinde sadece karşılıklı haberleşen birimlerin kullandıkları şifrelerin değiştirilmesinin yanı sıra kullanılan şifreleme algoritmasının da

değiştirilmesi hatta gönderilen verinin veya kullanıcıların özelliğine göre kademeli şifre kullanılması bu alanda kullanılabilir yaklaşımlardandır.

3. İlgiye Yönelik Programlama

Nesneye Yönelik Programlama (NYP) yaklaşımı özellikle dağıtılmış yazılım geliştirmekte önemli bir rol oynamaktadır. Ancak yazılımların boyutu büyüdükçe ve karmaşıklıkları arttıkça, sistem içerisinde kullanılan nesne sayısı artmakta, farklı nesnelere arasında ortak işlevlerin olduğu görülmektedir. NYP bu ortak işlevleri uygun bir modülerlikte tek bir modül/sınıf içerisinde gerçekleştirmekte yetersiz kalmaktadır.

Bu noktada son yıllarda üzerinde yoğun bir çalışma yürütülen İYP yaklaşımı öne çıkmakta ve bu gibi karmaşıklaşmış kodlar içerisinde, enine kesen ilgi olarak tanımlanan, işlevlerin modüler ve anlaşılır bir hale getirilmesine olanak sağlamaktadır. 2001 yılında Massachusetts Institute of Technology (MIT) tarafından İYP yaklaşımının gelecekte dünyayı etkileyecek ve değiştirecek 10 anahtar teknolojilerden biri olacağını ilan etmesi [3] İYP alandaki çalışmaları yoğunlaştırmıştır.



Şekil 2: İlgiye Yönelik Programlama Yaklaşımı

İYP yaklaşımında yazılım mimarisi Şekil 2’de görüldüğü gibi iki parçaya bölünmüştür: *Ana Sistem* ve *İlgi Sistemi*. *Ana Sistem* fonksiyonel gereksinimleri karşılamakta olup genellikle Nesneye Yönelik yaklaşımla gerçekleştirilmiştir. Bunun yanında *İlgi Sistemi*, fonksiyonel olmayan işlevlerin farklı programlar/sınıflar üzerinde tekil bir yapıda gerçekleşmesini sağlamak üzere geliştirilmiştir. Her ne kadar işlevler farklı sınıflara dağıtılmış olsa da, her bir işlev “ilgi”³ olarak adlandırılan tek bir kaynaktan(modülde) toparlanabilmektedir. Bu sayede sistem NYP’nın sağlayamadığı şekilde işlevleri de modüler hale getirmek mümkün olmaktadır. Bu geliştirilen “ilgi”ler bir *ilgi işleyici*⁴ tarafından sisteme işlenmekte ve Ana Sistemdeki gerekli sınıf yapıları uygun şekilde değiştirilmektedir. Bu işlemlerin sonucunda bilgisayar üzerinde çalışacak son sistem üretilmektedir (bakınız Şekil 2).

Zaman içerisinde ilgilerde veya sınıfların fonksiyonelliklerinde bir değişiklik yapılması gerektiğinde değişiklikler Ana Sistem veya İlgi Sistemi üzerinde yapılır ve daha sonra İlgi İşleyici ve derleyici üzerinden yeniden son sistem oluşturulur.

İlgiye Yönelik yaklaşım Java, C++, C#, Smalltalk, Ruby, Scheme ve benzer bir çok programlama diliyle beraber kullanılmaktadır. Her ne kadar programlama dilleri doğrudan bu yaklaşımı desteklemese de kullanılan yardımcı araçlar sayesinde İlgiye Yönelik bir yazılım mimarisi geliştirmek mümkün olmaktadır. Bu yardımcı araçlardan en çok kullanılanlarından birisi sağlamış olduğu bir çok avantajdan dolayı Java programlama dilinin destekleyen AspectJ derleyicisidir.

3.1. AspectJ

İlk sürümü 2001 yılında AOP Xerox PARC İYP grubu tarafından geliştirilen yazılım bu alanda en çok tercih edilen arabirimlerden biri olup, internet üzerinden kolaylıkla temin edilebilmekte ve güncellemeleri sürekli olarak yapılmaktadır.

AspectJ Java programlama dili ile uyumlu çalışmakta ve yazılım geliştiricilere tam bir İYP desteği sağlamakta olup, geliştirilmiş olan ilgileri ana mimarideki sınıflar ile birleştirmek için ajc (aspect java compiler) derleyicisini kullanmaktadır.

Bir AspectJ uygulamasında herşey kesme noktaları(join points) üzerinde cereyan etmektedir. Bu noktalar programın içerisinde ilgilerin uygun şekilde işlenecekleri yerleri ifade etmektedir. AspectJ terminolojisine göre tanımlanan örnek bir ilgi(aspect) yapısı Şekil 3’te gösterilmekte olup iki farklı türde enine kesen ilgi tanımlanabilmektedir.

```
public aspect ExampleAspect {

// Dinamik Enine Kesme
pointcut doSomething()
call (* ExampleClass.do*(..));

before() : doSomething() {
... advice body
}

after() : doSomething() {
... advice body
}

// Statik Enine Kesme
declare warning : <pointcut> : <message>;
declare error : <pointcut> : <message>;
declare precedence : Aspect1, Aspect2;
declare parents : [Child] extends [Class]
private float ExampleClass. dataMember;
}
```

Şekil 3. AspectJ İlgi Yapısı

³ Aspect

⁴ Aspect Weaver

Statik Enine Kesen İlgi sistemin çalışma davranışlarını etkilemeyerek sadece yeni tanımlamalar eklemektedir. Bu sayede ilgili sınıfa yeni durum/sınıf değişkenleri eklemek ve derleme zamanında bazı hata kontrolleri yapabilmek olanağı sağlanmaktadır.

Dinamik Enine Kesen İlgi ise doğrudan uygulamanın davranışları üzerinde etkide bulunmaktadır. Bu tip ilgi ile kontrol yapılacak kesme noktaları POINTCUT lar ile uygun koşullar belirlenerek seçilmekte ve ADVICE lar ile de ilgili kesme noktasının öncesinde (before), sonrasında (after) veya etrafında (around) belirli program komutları bir altprogram şeklinde çalıştırılabilmektedir.

4. İlgiye Yönelik Güvenlik Modeli

Güvenlik modeli İYP yaklaşımı ile daha öncede birkaç kez gerçekleştirilmiştir. Bu sayede sistem üzerindeki güvenlik ile ilgili program parçaları tek bir modül içerisinde (*Güvenlik İlgisinde*) toparlanabilmiştir. Bu sayede eğer güvenlik yaklaşımı değişir ise sadece güvenlik modülü içerisindeki ilgili yapının değiştirilmesi yeterli olacaktır. Bu sayede güvenlik modülünü diğer program parçalarından ayırarak konunun uzmanı yazılımcılar tarafından geliştirilmesine de

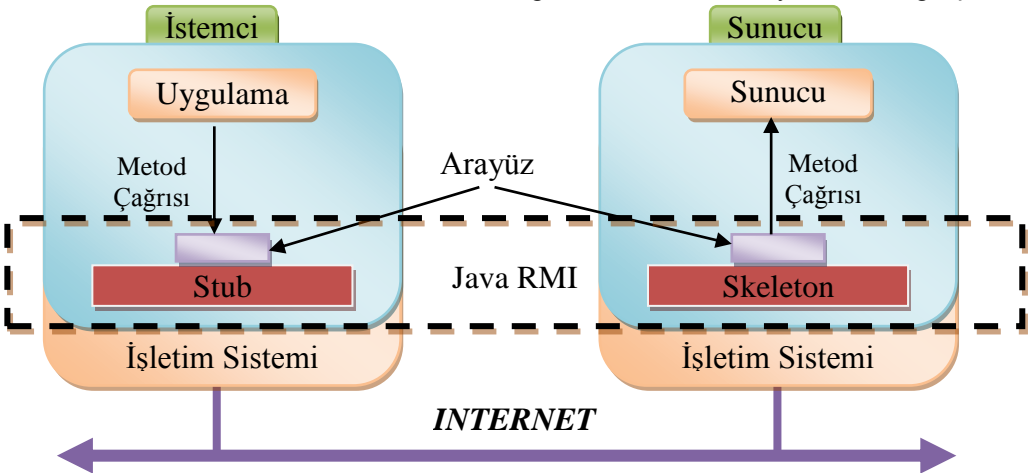
olanak sağlanmaktadır.

Michiaki Tatsubori vekil sınıfların dağıtılmış ortamda kullanımı ile ilgiye yönelik programlamayı destekleyen bir araç geliştirmiş bu sayede programcılarının sistemde dağıtılmış olan ilgileri tanımlayabilmesine olanak sağlamıştır. [4]

Henner Jakob ve arkadaşları, geliştirilen bir dağıtılmış sistemde, sistem mimarisinin fonksiyonel olmayan işlevlerini (güvenlik ve servis kalitesi gibi) göstermek için *DiaAspect* olarak tanımladıkları bir İlgiye Yönelik bir mimari tanımlama dili geliştirmişlerdir. Bu dil sayesinde ilgiler arasındaki koordinasyonun tam olarak ortaya konması amaçlanmaktadır [5].

Bazı araştırmacılar ise Dağıtılmış Sistemler geliştiriminde biraz daha Java RMI haberleşen arabirimler üzerine yoğunlaşmış ve ilgiye yönelik yaklaşımla bu kesişen işlevlerin modüler bir şekilde kodlanmasında akıllı vekil nesnelere kullanılması hususunu araştırmışlardır [6-7].

Haberleşme protokolü dağıtılmış sistemlerde çok temel bir işlev olup, bu işlevin güvenlik gereksinimlerinin yazılım geliştirme



Şekil 4: Java RMI Haberleşme Modeli

sürecinin başlangıcında belirlenmesi beklenir. Ama gelişen süreç dâhilinde güvenlik ihtiyaçları ad-hoc bir yapı gösterir ve sürekli olarak değişebilir. Farklı protokollerin gerçekleşmesine veya farklı şifreleme algoritmalarının kullanılmasına ihtiyaç duyulabilir. Ancak geniş kapsamlı dağıtılmış yazılım sistemlerinde bu güvenlik ihtiyacı sistemin birçok yerinde kullanılması gerektiği için tek bir model içerisinde bulunmayıp sistem içerisinde farklı sınıflarda ayrı ayrı gerçekleşmiş olarak kullanılır. Bu yapı modülerliği engel teşkil etmekte ve yazılımın güncellenmesinin ve anlaşılabilirliğini zorlaştırmaktadır.

5. Önerilen Yazılım Mimarisi

Önerilen yazılım mimarisi, daha önceden geliştirilmiş olan bir dağıtılmış mimarinin güvenilir haberleşmesini sağlamak üzere tasarlanmıştır. Önceden geliştirilmiş olan sistem Şekil 4'te gösterildiği gibi Internet üzerinde güvenli olmayan bir yapıda haberleşmektedir. İYP yaklaşımı kullanarak hem sistemin daha modüler hale getirilmesi hemde daha gürbüz olmasının sağlanması amaçlanmaktadır. Önceden geliştirilmiş sistemde kontrol edilmesi gereken noktalar Java RMI sınıfları ve komutları ile alakalıdır. Sistemde temelde yapılması gereken 4 ana işlev bulunmaktadır.

- Kontrolsüz bir şekilde RMI bağlantısının kurulmasının engellenmesi
- RMI bağlantılarının İlgiye Yönelik olarak yapılması
- Uzaktaki nesneye gönderilen mesajların şifrelenmesi (modüler yapıya uygun olarak tek bir ilgi üzerinde)
- Uzaktaki nesneye şifreli gelen mesajların şifresinin çözülmesi (modüler yapıya uygun olarak tek bir ilgi üzerinde)

Yukarıda bahsedildiği gibi AspectJ, statik enine kesen ilgi yaklaşımı ile istenen ek hata

kontrolleri yapabilme imkânına sahiptir. Bu nedenle öncelikli olarak kontrol dışı bir şekilde RMI bağlantısının kurulması AspectJ nin özel komutları sayesinde şu şekilde sağlanabilir.

```
declare error
: call(void Naming.rebind(..))
|| call(void Naming.bind(..))
|| call(void Naming.lookup(..))
&& !within(Guvenlik)
: "RMI baglanti islemleri sadece ilgili Aspect tarafından
cagrilabilir.";
```

Artık bir dağıtılmış mimari geliştirirken doğrudan RMI bağlantısı sağlanamayacağı için RMI bağlantılarının tek bir elden ve uygun bir İlgi yapısında düzenlenmesi gerekmektedir. Bunun için mimaride bağlantı yapılacak yerlere ihtiyaç duyulan birleşme noktaları (join point) tanımlanır. Uygun bağlantının sağlanabilmesi için bazı parametrelerinde ilgiye gönderilmesi gerekmektedir. İhtiyaç duyulan tanımlama mimari içerisinde şu şekilde yapılır.

```
public class Client extends
    UnicastRemoteObject implements *** {
public Client() throws RemoteException {
    super();
    **** }
public void CONNECT(String str, Client client)
    throws Exception {
    Gerekli RMI Bağlantı Komutları
    *****
}
*****
}
```

Bağlantı noktasında çalıştırılacak ilgi komutları şüphesiz RMI bağlantılarını tanımlaması gerekmektedir. Bu tanımlama geliştirilen Güvenlik ilgisi içerisinde şu şekilde tanımlanır.

```
pointcut connect(Client client, String name, Client obj)
:call(void Client.RMIConnect(String, Client))
&& target(client)
&& args(name,obj);
```

Sunucu nesneye mesaj gönderme metodlarına uygun olarak tek tek bağlantı noktaları tanımlanır ve ilgili tavsiye metodları gerçekleşir. Örneğin eğer sunucu nesnenin send(String) metodu çağrılarak mesaj gönderiliyor ise ilgili metod çağrılarını üzerine uygun bağlantı noktalarının aşağıdaki gibi tanımlanması

Şifreleme ve şifre çözme ilgileri Güvenlik İlgişi içerisinde şu şekilde tanımlanır.

```
public String encrypt(String classname, String
methodname, String data){
    Sınıf Adı ve Metod Adına göre Grup Şifresini Al()
    String Veriyi Grup Şifresi ile Şifrele()
    Şifreli Stringi dönder
}

public String decrypt(String classname, String
methodname, String data){
    Sınıf Adı ve Metod Adına göre Grup Şifresini Al()
    String Verinin Grup Şifresi ile Şifresini Çöz()
    Stringi dönder
}
```

Sunucu ile bağlantıyı sağlayan arayüzün ilgili metodunun (send metodunun) çağrılması ile yukarıda gösterilen şifreleme metodu çağrılarak ilgili veri şifrelenir. Daha sonrada aşağıda gösterilen Kesme Noktası ve Tavsiye yapısına uygun olarak sunucu nesnesine gönderilir.

```
pointcut send(Server_Int server, String data)
: call(void Server_Int.send(String))
&& target(server)
&& args(data);

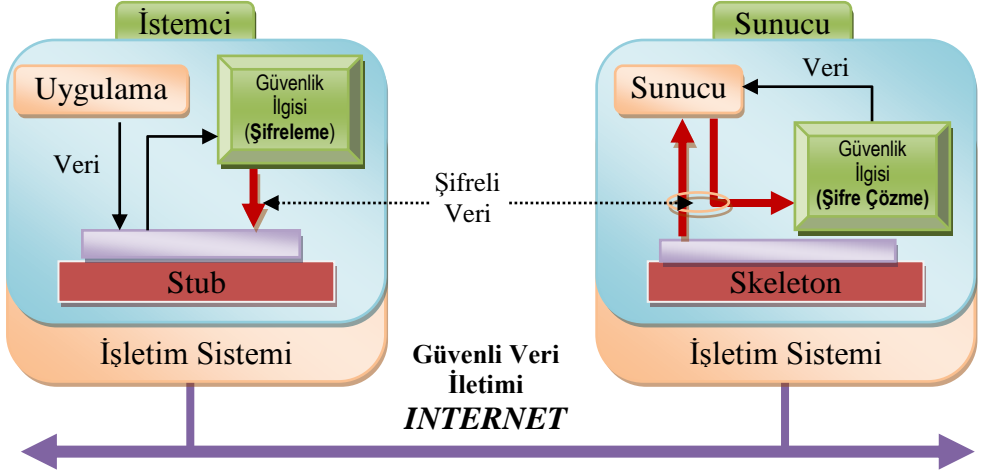
void around(Server_Int server, String data)
: send(server, data) {
    Veriyi Şifrele()
    Şifreli Veriyi Gönder()
}
```

Şifreli verinin sunucu tarafından da yeniden şifresinin çözülmesi ve gerekli işlemlerin yapılması da *Güvenlik* ilgisi içerisinde halledilir.

Bu sayede güvenli veri iletimi yapmayan bir dağıtılmış yazılım mimarisinin İYP yaklaşımı ile kontrollü ve güvenli bir şekilde veri iletmeye olanak sağlanmış olmaktadır. Şifreleme ve şifre çözme işlemleri Şekil 5'te görüldüğü gibi tek bir modül içerisinde toplandığı için istendiği gibi güncellenmesi ve değiştirilmesi olanağı da sağlanmaktadır. Transfer edilecek verilerin şifrelenmesinde simetrik şifreleme algoritmalarından olan DES, AES, Blowfish, Twofish, RC4, 3DES, IDEA ve benzeri algoritmalar kullanılabileceği gibi Yazılımcı kendi özel şifreleme algoritmasını da geliştirerek kullanma olanağına sahiptir. Özellikle String veri tipi haricindeki basit veri tiplerinin güvenli transferinde bu tip algoritmaların kullanılmasında fayda vardır.

Karşılıklı haberleşen nesnelere arasında transfer edilen verinin bir byte dizisi(byte[]) olarak tanımlanması ise bize serileştirilebilecek her türlü nesnenin (Serializable arayüzünü gerçekleştiren) güvenli olarak iletilmesine olanak sağlayacak ve sistemin esnekliği artıracaktır. Ancak amaç daha önceden geliştirilmiş ve aktif kullanılan bir sistem üzerinde İlgiye Yönelik yaklaşım ile bir güvenli haberleşme ortamı eklemek olduğu için Basit veri tiplerinin iletimindeki şifreleme işlemleri ön planda incelenmiştir.

Nesneler arası mesajların güvenli şekilde gönderilmesi için simetrik şifreleme algoritmalarından faydalanılır. Simetrik şifreler Bir Anahtar Dağıtım Merkezi(ADM) tarafından gerekli asıllama işlemleri yapılarak haberleşen nesnelere dağıtılır. Haberleşen nesnelere arasında doğrudan gizli ve açık anahtarlarla da şifreleme yapma imkânı vardır. Ancak haberleşen birimlerin önceden belirli olmadığı durumlarda, dinamik olarak yeni aktif nesnelere eklenebileceği ve çıkarılabileceği durumlarda, bu asimetrik anahtarların önceden temin edilmesi mümkün



Şekil 5: İlgiye Yönelik Yaklaşım ile Java RMI Üzerinden Güvenli Veri İletimi

değildir. Aynı zamanda asimetrik anahtarlarla yapılan şifreleme daha fazla süre almakta ve gerçek zamanlı veri iletişimde gecikmelere yol açmaktadır. Bu nedenle ihtiyaç duyulan anahtarların bir ADM üzerinden dağıtılması daha uygun bir yaklaşım olarak görülmektedir. Burada sistem üzerinde bir oturum (session) anahtarlaması yapılmakta ve belirli süre sonra bu anahtarlar değiştirilmektedir.

Bu şekilde kullanılan ADM yapısı sayesinde Sınıf isimlerine bağlı veya her sınıfın içerisinde ayrı ayrı metodlara bağlı olarak Grup Anahtarları kullanılabilir. Grup anahtarlarının da dağıtılması ve güncellemesi ADM'nin sorumluluğunda gerçekleştirilebilir. Bu şifre dağıtım mekanizmasının yapısı ileriki çalışmalarda anlatılması planlanmaktadır.

6. Sonuç

Dağıtılmış programlamada güvenli haberleşmenin sağlanması karmaşık ve zor bir konudur. Özellikle dağıtılmış nesnelerin sayısı arttıkça, bu nesneler arasındaki veri alışverişinin haricinde şifrelerinde güvenilir bir şekilde dağıtılması gerekmektedir. Hâlbuki güvenlik işlevi genellikle mimari

üzerindeki farklı nesneler üzerine dağılmış bir işlev olup, enine kesen bir yaklaşım sergilemektedir.

İYP metodoloji bu enine kesen ilgileri tek bir modül içerisinde birleştirme olanağı vermekte olup, bu metodolojinin dağıtılmış yazılım geliştirmede kullanılması bu tip sistemlerde modülerlik, anlaşılabilirlik, sürdürülebilirlik, genişleyebilirlik, yeniden kullanılabilirlik, izlenebilirlik, esneklik ve kolay geliştirilebilme gibi birçok yazılım kalitesinin artırılmasını sağlamaktadır. Bunların sonucu olarak gerek yazılım geliştirme süresi kısalmakta gerekse yazılım geliştirme masrafı ise azalmaktadır.

Bu nedenle İlgiye Yönelik Programlamanın dağıtılmış sistem geliştirmede kullanılması bize hem daha modüler bir yapı oluşturmayı sağlayacak, hem de daha önce geliştirilmiş olan bir dağıtılmış sistemi güvenli haberleşebilen bir hale kolaylıkla getirmemize imkân verecektir. Bu bildiride önceden geliştirilmiş olan ve Java RMI yapısı üzerinde güvenli haberleşme yapmayan bir dağıtılmış sistemin, İYP yaklaşımı ile güvenli haberleşmesinin sağlanması için yapılması gerekenler sıralanmıştır.

Kaynaklar

1. Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, John Irwin: Aspect-Oriented Programming. European Conference on Object-Oriented Programming ECOOP 1997: 220-242.
2. Tzilla Elrad, Mehmet Aksit, Gregor Kiczales, Karl Lieberherr and Harold Ossher, Discussing Aspects of Aspect-Oriented Programming AOP: Frequently-Asked Questions. Communications of the ACM, 44 (10). pp. 33-38, 2001.
3. Terry J. Van Der Werff. 10 Emerging Technologies That Will Change The World. Technology Review, 2001.
4. Michiaki Tatsubori, Separation of Distribution Concerns in Distributed Java Programming, In Addendum to the 2001 Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2001), Doctoral Symposium, pp.19-20, Tampa Bay, Florida, USA, October 14-22, 2001.
5. Henner Jakob, Nicolas Lorient, and Charles Consel. 2009. An aspect-oriented approach to securing distributed systems. In Proceedings of the 2009 International Conference on Pervasive Services (ICPS '09). ACM, New York, NY, USA, 21-30.
6. Inderjit Singh Dhanoa, Er.Dalwinder Singh Salaria, Aspect Oriented Java RMI Server, INDIACom-2010, Computing For Nation Development, New Delhi, February 25 – 26, 2010
7. A. Stevenson, S. MacDonald, "Smart proxies in Java RMI with dynamic aspect-oriented programming," *IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008.*, pp.1-6, April 2008