Farklı Yazılım Teknolojilerinde Mobil Kod Güvenliğini Sağlama Yöntemleri

Ömer Özgür Bozkurt[1], Göksel Biricik[2]

[1,2] Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, 34349, Beşiktaş, İstanbul
[1] ozgur@ce.yildiz.edu.tr, [2] goksel@ce.yildiz.edu.tr

Özet. Yeni uygulama geliştirmede en büyük itici güç mobil kod kullanımı olarak karşımıza çıkmaktadır. Geliştirilen bu kod parçaları, sistemleri iki açıdan korunmak zorunda bırakmaktadır: İşletim ortamının kötü niyetli kodlardan korunması ve ana sistemlerin kötü niyetli kodlardan korunması. İnternet'teki kullanımları da dikkate alınırsa, mobil kodlarda güvenliğin ne kadar ciddi bir problem olduğu kendiliğinden ortaya çıkmaktadır. Başka bir deyişle, zararlı koddan korunmak bir gereksinim olduğu kadar kodun korunması da bir gereksinimdir. Bu bildiride; kullanıcı, sistemler ve yazılımcı açısından yazılımı ve kendilerini korumak için yapılabilecekler ayrıntılı olarak incelenmektedir.

1 Giriş

Günümüzde mobil kod teknolojisi bilgisayar ağları ve dağıtık sistemlerde yeni uygulama geliştirimi için itici güç olmuştur. Kodun mobil olması, bir programın tamamının ya da parçalarının bilgisayar sistemleri arasında alışverişinin yapılması ve işletim ortamı değişkenliğinin program ya da program parçacıklarının yazıldığı programlama dili tarafından gizlenmesi anlamına gelmektedir.

Mobil kod kavramı, 1980lerin başlarında homojen bir yerel ağ ortamında nesne hareketliliğini getiren Emerald [9] ile ortaya çıkmıştır. Ardından R2D2 [11] ve Chorus [7] uğradıkları düğümlerde işletilecek kodu taşıyan ve tüm ağı dolaşan "aktif mesajlar" kavramını kullanmıştır. 1990 yılında ortaya atılan REV (Remote Evaluation – Uzak Değerlendirme) mekanizması [10] istemcinin yordam kodu ve parametrelerini sunumcuya göndermesine olanak sağlamamaktaydı. Bu noktadan sonra iki tür mobil kod teknolojisinden söz etmek mümkündür. [4]

Güçlü mobil teknolojiler; işletilmekte olan bir programın tamamının ya da parçacıklarının işletim kesilerek, program durumu ve verileriyle beraber bir başka ortama aktarılması ve bu ortamda işletime devam etmesini sağlar. Örnek olarak 1990ların başında General Magic tarafından ajan tabanlı ağ programlama dili olarak geliştirilen Telescript [12] ve ardından gelen araştırma amaçlı Tacoma [8] ile Tcl dilini mobil hale getirmek için kullanılan Agent Tcl [14] sayılabilir.

Zayıf mobil teknolojiler; işletilecek kodun (ve belki bir takım başlangıç verilerinin) bir bilgisayardan diğerine aktarımının sağlanarak işletimin bu ortamda yapılmasını ifade eder. Bu türe en yaygın örnekler Microsoft tarafından geliştirilen ActiveX ile Sun Systems tarafından geliştirilen Java teknolojileridir.

Mobil kod sistemleri, yapısı gereği hem kod hem de işletim ortamı tarafında güvenliğe gerek duymaktadır [13]. Bu da, güvenlik probleminin iki taraflı ele alınması ve çözülmesi gerektiği anlamına gelmektedir. Bu bildiride işletim ortamının tehlikeli koddan ve mobil kodun kötü niyetli işletim ortamından korunması için geliştirilen yöntemler ele alınmaktadır. Mobil kod imkanı sağlayan farklı yazılım teknolojilerinin genel yapısı ve kullandıkları güvenlik yöntemleri, açıklarıyla birlikte incelenmektedir.

2 İşletim Ortamının Korunması

Mobil kod güvenliği ile ilgili belki de en belirgin sorun, kodun çalıştırılması düşünülen işletim ortamının güvenliğidir. Ortamın çalışmakta olan mobil kodun zararlı işlemlerinden korunması, ciddi bir sorun oluşturmaktadır.

Mobil kodun, truva atları, bilgisayar virüsleri, ağ solucanları gibi potansiyel tehlike içerdiği düşünülür. Teoride, bu kötü amaçlı kodlardan korunmak, oldukça zordur. Bir mobil kodun, çalıştığında zararlı olabileceğini belirleyebilecek bir yöntem bulunmamaktadır. İşletim ortamının korunması, bu yüzden özellikle üzerinde durulması gereken bir konudur.

İşletim ortamının kötü niyetli mobil koddan korumak için, üç yöntem geliştirilmiştir. Kum havuzu yöntemi (sandboxing), sayısal çek-kapla yöntemi (digital shrink-wrapping) ve taşınan kodun kanıtlanması yöntemidir (proof-carrying code). Farklı yazılım teknolojilerinde işletim ortamının güvenliğinin sağlanması için bu yaklaşımlar tek başlarına kullanılabilecekleri gibi, birleştirilerek de kullanılabilirler. [1], [5], [13]

2.1 Kum Havuzu Yöntemi (Sandboxing)

İşletim ortamını, mobil kodun çalışmasından doğabilecek tehlikelerden korumanın bir yolu, kodun ortama erişim izinlerinin kısıtlandığı bir ortamda çalıştırılmasıdır. Mobil kod, sadece belirlenen özelliklere ulaşabileceği bir kum havuzu (sandbox) içerisinde çalıştırılır. Örneğin, kodun sadece ekrana çizim yapabilmesine izin verilirken, yerel dosya sistemine ve ağ kaynaklarına erişimi kısıtlanır. Bu yöntem uygulandığında, mobil kodun çalışmasına çok sıkı

kısıtlamalar getirilmektedir. Belirli uygulamalar için (dosya erişimi yapması gereken metin editörü gibi), bu kısıtlamalar kodun gerçekleştirmesi gereken işlemi yapmasını engeller. [1], [5]

2.2 Sayısal Çek-Kapla Yöntemi (Digital Shrink-Wrap)

Çalışma ortamını zararlı mobil koddan korumak için geliştirilmiş olan bir yöntem de, mobil kodun çalıştırılmadan önce onaylandığı sayısal çek-kapla yöntemidir. Mobil kodun çalışması sırasında yapacakları önceden kestirilemese de, en azından güvenli bir kaynaktan geldiği doğrulanabilir. Kod sağlayıcı, mobil kodu sayısal olarak imzalar ve uygun üretici sertifikası ile birlikte dağıtır. Kullanıcı, bu yöntem ile kodun sağlayıcısına duyduğu güvene bağlı olarak kendi işletim ortamında mobil kodu çalıştırmayı kabul veya reddedebilir. [5], [13], [15]

2.3 Taşınan Kodun Kanıtlanması Yöntemi (Proof-Carrying Code)

Potansiyel tehlike içeren mobil kodu korumak için geliştirilen yöntemlerden olan taşınan kodun kanıtlanması yöntemi, başka bir sistem tarafından sağlanan mobil kodun kurulması ve çalıştırılmasının güvenli olduğunu, otomatik olarak ve doğrulukla belirlemeyi hedefler. Bu teknikte kod sağlayıcı, kendi mobil kodunun, istemcinin belirlediği güvenlik ilkelerine (security policy) uygun olduğunu belirten bir kanıt belirtir. Bu kanıt, sayısal olarak kullanıcıya iletilir ve belirlenen bir doğrulama işleminden geçer. Taşınan kodun kanıtlanması yöntemindeki en önemli kısıt, işletim ortamına sahip olan kullanıcının belirlediği güvenlik ilkelerinin doğru ve sağlayıcı tarafından anlaşılabilir şekilde olmasıdır. [5]

3 Mobil Kodun Korunması

İşletim ortamının, çalışan bir mobil koda saldırması geleneksel bilgisayar güvenliğinde az görülen bir durumdur. Bunun sebebi, işletim ortamını bulunduran ortamın genellikle programı da barındırmasıdır. Ancak mobil kod söz konusu olduğunda, mobil kod sağlayıcı ve bu kodu işleten taraflar, birbirinden ayrıdır. Bu durum, tehlikeli ortamlardan mobil kodun korunmasını gerektirmektedir. Tehlikeli ortam, başka bir sağlayıcının ürettiği mobil kodu bünyesinde barındıran ve bu koda saldırmaya çalışan, veya bu kodu kullanarak başka işletim ortamları için tehlike yaratan birim olarak tanımlanabilir. Tehlikeli ortam, mobil kodun veri yapısını, kontrol akışlarını incelemeye ve örneklemeye çalışabilir, başka işletim ortamlarını bu kod ile yanıltabilir ya da bu kodun yaptığı sistem çağrılarına yanlış yanıtlar verebilir.

Mobil kodun işletim ortamından korunması normalde beklenen bir durum olmasa da, özellikle ajan tabanlı sistemlerde büyük önem kazanmaktadır. İşletim ortamı, mobil kodu üzerinde çalıştıracağı için, veri yapısı ve akışı üzerinde tam kontrole sahiptir. Bu nedenle mobil kodun işletim ortamından etkili bir biçimde korunması çok zordur. Bu da, mobil kodun çalışacağı ortama güvenmesi gerektiğini ortaya koymaktadır. Diğer bir deyişle, mobil kod teknolojisi, ancak güvenilir ortam tabanında gerçekleştirilebilir.

İşletim ortamlarını barındıran kötü niyetli birimlerden mobil kodu korumak için çeşitli çalışmalar yapılmıştır. Sonuçlar, aşağıda açıklanan dört grup altında toplanabilir:

- Mobil kod, güvenilmeyen ortamlara çalıştırılmak üzere gönderilmez. Buradaki sorun, bir işletim ortamının güvenilir olup olmadığının belirlenmesidir. Güvenilir ortamı belirleyebilmek için kesin bir karar mekanizması kurmak çok güçtür.
- İkinci yaklaşım, örgütsel bir çözüm getirmektedir. İşletim ortamları, gelişigüzel oluşturulamaz, ancak güvenilir birimler mobil kod çalıştırabilecek işletim ortamları barındırabilirler.
- Mobil kodun bütünlüğünü bozabilecek etkenlere karşı, işletim ortamlarında fiziksel, özelleştirilmiş, tetkiklere dayanıklı yapıların bulunması gerekir. İşletim ortamını barındıran her birimde böyle bir yapının bulunması hem çok zor, hem de fazlasıyla kısıtlayıcıdır.
- Mobil kodu korumak için kısıtlı işletim ortamları olmalı ve mobil kod ve ortam arasında şifreli bir protokol bulunmalıdır. Bu şifre temelli koruma sayesinde, mobil kod üzerinde oynamalar yapmanın önüne geçilebilir.

Bu yaklaşımların çoğu, çok kısıtlayıcı olmalarından ya da fazla güvenilir olmamalarından, yetersiz kalmaktadırlar. Güncel çözümler, işletim ortamından gelen saldırıları belirleme veya önlemeye yöneliktir. Önerilen çözümlerden kısıtlı karakutu güvenliği ve şifrelenmiş işlevlerle hesaplama yöntemleri saldırıları önlemeye yönelikken, şifresel izleme yöntemi, saldırıları belirlemek için geliştirilmiştir. [1], [5], [13], [15]

3.1 Kısıtlı Karakutu Güvenliği (Limited Blackbox Security)

Mobil kodu tehlikeli ortamdan koruma yöntemlerinden olan kısıtlı karakutu güvenliğinin ana fikri, kodu değiştirme ve okuma tehditlerinden koruyacak tanımlamalara uygun çalıştırılabilir kod yaratmaktır. Mobil kod, bir karakutudur ve iç yapısı değil, sadece girdi ve çıktıları gözlemlenebilir. Ancak, bu tip bir karakutu güvenliğini sağlayabilecek bir algoritma geliştirilememiştir.

Karakutu güvenliğini sürekli sağlamak imkansız olduğu için, yöntemde sadece belirli zaman dilimi için koruma gerçekleştirilir, bu süre içinde koda yapılan saldırılar karşılanır. Bu zamanı tam olarak belirleyebilmek için, kodun güvenlik süresinin bitişi karakutuya eklenebilir. Mobil kodun işleyişi, saldırgan tarafından incelenebilir. Ancak, kodun çalışmasının çözülebilmesi, belirli bir zaman alır. Kodun içinde karıştırıcı işlemler ve algoritmalar kullanılarak, yapının çözülmesinin normalden daha uzun zaman alması sağlanır. Bu süre içinde de, karakutu güvenliği sağlanmış olur.

Kısıtlı karakutu güvenliği, her saldırıya karşı koruma sağlayamamaktadır. Örneğin tehlikeli ortam, her zaman için kodun sistem çağrılarına yanlış cevaplar üretebilir. Ayrıca, sonuçları belirsiz de olsa, saldırgan kodda ve veri yapılarında değişiklik yapmaya çalışabilir. Uygulamaların tiplerine uygun olarak güvenli çalışma zamanlarının belirlenmesi, bu yöntemin gerçekleştirilmesinde karşılaşılan en büyük zorluktur. [5]

3.2 Şifrelenmiş İşlevlerle Hesaplama (Computing With Encrypted Functions)

Mobil kodun tehlikeli ortamdan korunması için bir yöntem de, şifrelenmiş işlevlerle hesaplamadır. Mobil kod, içeriğinin ve çalışma şeklinin bilinmemesini istediği bir işlevi, şifreleyerek çalıştırtır. Oluşan sonucu da, şifreyi çözerek kullanır. İşletim ortamı, kodun nasıl çalıştığı ve neler yaptığını izleyemez.

Bu yöntemin etkin çalışması için doğru şifreleme yöntemlerinin bulunması gereklidir. Mobil şifreleme, mobil çalıştırılabilir kod için uygun matematiksel yöntemlerin bulunmasıyla gerçekleştirilebilir. Şifrelenmiş işlevlerle hesaplama yöntemi, mobil kodun işletim ortamından korunması için kesin bir çözüm getirmese de, karakutu yönteminin gerçekleştirilebilir olduğunu ortaya koymaktadır. [1], [6], [13]

3.3 Şifresel İzleme Yöntemi (Cryptographic Traces)

Bu yöntem, önceki yöntemlerin aksine, mobil kodun içeriğine, durumuna ve kontrol akışına yapılan saldırıları belirlemeye yarar ve bunun için takipsel ve şifresel yöntemler kullanır. Şifresel izleme yöntemi, kodun durumuna, akışına yapılan herhangi bir geçersiz değişikliği belirlemeye çalışır. Mobil kodun çalışması süresince oluşan verilerin takip edilmesi tekniğine dayanır. Bu izleme bilgileri, mobil kodun çalışmasının onaylanması için kullanılır. Herhangi bir değişiklik girişiminde, mobil kod sahibi iddia edilen işlemlerin kod tarafından asla gerçekleştirilemeyeceğini kolaylıkla ispatlayabilir. [5], [6], [13]

4 Mobil Kod Sistemleri ve Güvenlik Özellikleri

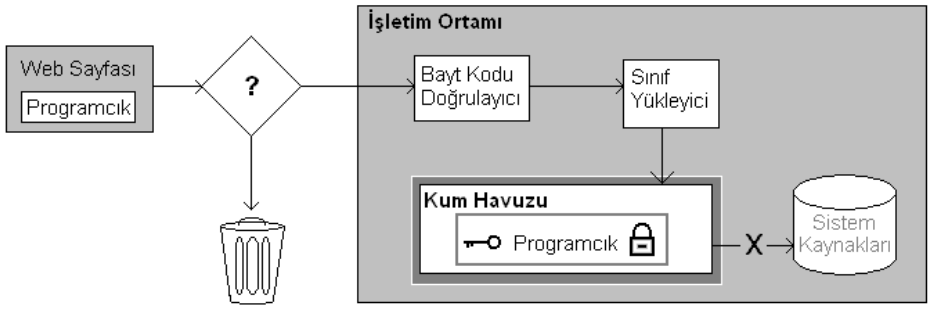
En yaygın kullanılan ve neredeyse tüm İnternet kullanıcılarının aşina olduğu mobil kod teknolojileri olan Java ve ActiveX tek bir noktada benzeşmektedir; kodun indirilmesi ve uzaktan işletimi. Java hem bir işletim ortamı hem de bir programlama dilidir. Java programlama dilinin sınırları kesin olarak çizilmiş ve “iyi huylu” olmak üzere tasarlanmıştır. Devam eden kısımlarda, Java ve ActiveX mobil kod güvenliği açısından getirdikleri ve sorunları tartışılmaktadır.

4.1 Java

Sun Microsystems tarafından geliştirilen ve 1995 yılında kullanıma sunulan Java, mobil kod konusunda getirdiği özellikler ile oldukça dikkat çekmiş ve yaygınlaşmıştır. Java programlama dili tasarlanırken esas amaç, taşınabilir, kolay öğrenilen, genel amaçlı, platform bağımsız bir nesneye dayalı programlama dili yaratmaktı. Ancak dilin sunduğu özellikler ve İnternet’in yaygınlaşması ve büyümesi ile, bugün en çok bilinen ve kullanılan mobil kod teknolojilerinden olmuştur. [5]

Java derleyicisi, kaynak kodlarını platform bağımsız bir ara dil olan Java Bayt Kodu’na (Java Byte Code) dönüştürür. Bu kod, daha sonra her platformda bulunan Java Sanal Makinesi (Java Virtual Machine) aracılığıyla işlenerek, çalıştırılır. Sanal makine, mobil kodun işletim ortamını oluşturur. [1]

Java, mobil kod alanındaki asıl başarısını, İnternet’le tümleşik olarak çalışabilmesine borçludur. Java işletim ortamına sahip olan tarayıcılar, yaygın olarak bulunmaktadır ve Java programcıklarını (applet) çalıştırabilmektedirler. Ayrıca Java işletim ortamı, yaygın kullanılan işletim sistemlerinde de desteklenmektedir. Java programcıkları, istemci tarafından çağrıldığında sunucudan alınarak işlenir ve çalıştırılırlar. Bu yüzden Java, zayıf mobilite sunmaktadır. [1], [5], [6]



Şekil 1. Java güvenlik mekanizmaları

İşletim ortamını tehlikeli mobil koddan korumak için Java, doğrulayıcı (verifier), sınıf yükleyici (class loader) ve güvenlik yöneticisinden (security manager) oluşan üç parçalı bir kum havuzu tekniği kullanmaktadır. Doğrulayıcı, mobil kodun gerçekten Java bayt kodu olduğunu ve sanal makinenin kurallarına uygunluğunu belirler. Sınıf yükleyici, yerel ya da uzak bir kaynaktan gelen doğrulanmış kodu sanal makineye yükler ve çalışmasını sağlar. Güvenlik yöneticisi ise, çalışmakta olan kodun yaptığı sistem çağrılarını ve erişmek istediği kaynakları kontrol altında tutar. Normal olarak, mobil Java programcıkları çok kısıtlı bir kum havuzu içerisinde çalıştırıldığından, ortamın mobil koddan yeteri kadar korunduğu söylenebilir. Kod, sistem için zararsız ancak rahatsız edici davranışlar yapabilir (Sürekli ses ya da ekran titreştirme gibi). [6], [13]

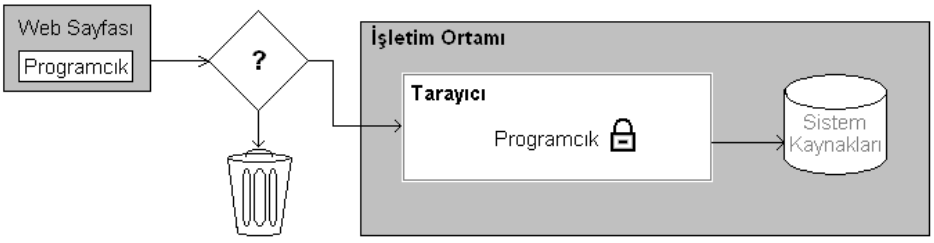
Java kodunun imzalanması ile de, mobil kodun güvenilir kaynaktan geldiği doğrulanabilir. Java, ayrıca her programcık için özelleştirilebilen, genişletilebilen erişim kontrolleri ve güvenlik politikaları ile de, işletim ortamını korumak için birçok önlem almıştır. Ancak, bunların ayarlanması genellikle karışık ve zordur. Ayrıca, karmaşıklık arttıkça performans düşmekte, güvenlik sisteminin yönetimi de zorlaşmaktadır. [1], [13]

Java programcıklarını kötü niyetli ortamlardan korumak için, Java ile birlikte gelen ve oldukça güçlü olan güvenlik uygulama programlama arabirimi (Security API) kullanılabilir. Şifreli işlevler kullanmak, kodun işleyişini korumak için kullanılan bir yoldur. Ayrıca, mobil kodun bilgi giriş çıkışları da programlama arabiriminin desteğiyle şifrelenebilir. Bu da, akan verinin yapısının kötü niyetli işletim ortamından korunmasını sağlar. Şifreleme yöntemleri dışında, Java'da herhangi bir kod güvenliği yöntemi bulunmamaktadır. [1], [6], [13]

Yukarıda açıklanmış olan işletim ortamının mobil koddan ve mobil kodun işletim ortamından korunması için Java'da kullanılan teknikler Şekil-1'de gösterilmiştir.

4.2 ActiveX

ActiveX, Microsoft tarafından belirli bir ortamda (ActiveX programı) ya da ağ üzerinde (ActiveX kontrol) işletilmek üzere her hangi bir programlama dilinde hazırlanan nesneyi "paketleme" olanağı sağlamak için geliştirilen teknolojidir. ActiveX temel olarak, kullanıcının bir tabloyu Microsoft Word dosyasına gömmesine olanak sağlayan teknolojiyi kullanır. Farklı uygulamalar arasında nesne gömmeye olanak sağlayan teknoloji, bir bilgisayar üzerindeki veri ve biçemi ortak kullanmayı amaçlayan Nesne Bağlama ve Gömme – Object Linking and Embedding (OLE) olarak adlandırılmaktadır ve ActiveX temelde bir OLEdir. ActiveX programları Microsoft'un Ortak Nesne Modeli – Common Object Model (COM) kullanılarak dağıtılmaktadır (Cert, 2000). Dağıtık Com (DCOM) kullanıcıların tek bir fare tuşuna basarak ActiveX kontrollerini indirmelerine ve işletmelerine olanak sağlamaktadır. ActiveX kontrolünün indirilmesi ve işletilmesi çok basit bir şekilde bir web sayfasındaki bağlantının tıklanması ya da bir elmek eklentisinin açılmasıyla gerçekleşebilir. ActiveX kontrol işletilmekte olduğu bilgisayarda yüklü bulunan tüm Windows programlarını ve sistem kodunu işletebilir. OLE tek bir bilgisayar üzerinde işletim için tasarlandığından güvenlik dikkate alınmamıştır. Bunun yanında bir teknolojinin ağ üzerinden erişilebilir hale getirilmesi tehdit ortamını genişletir ve OLEde rahatlık olarak değerlendirilen, ActiveXde açık haline gelir.



Şekil 2. ActiveX güvenlik mekanizmaları

İşletim sistemi tarafından sağlanan güvenlik ise, ActiveX programının işletildiği bağlama göre değişir. İndirilen ActiveX kontrol bağlamında tarayıcıyı işleten kullanıcının güvenlik bağlamıyla aynıdır. Windows 95/98/ME serisinde güvenlik kontrolleri bulunmamaktadır. Dolayısıyla bir ActiveX kontrolü kullanıcıyla herhangi bir etkileşime girmeksizin her şeyi gerçekleştirebilir. Windows NT tabanlı işletim sistemleri sistem kodu için donanım koruması ve işleyen görevler arasında izolasyonun yanı sıra kullanıcıların yetkilerini ve erişimlerini sınırlayan güvenlik önlemleri de sağlamaktadır. Şekil 2'de de görülebileceği gibi ActiveX kontrollerinin, kum havuzu benzeri hiç bir sınırlayıcı ortama bağımlı olmadıklarından, bilgisayarı kapatmaktan para aktarımına, bilgisayarın diskinin okunmasına ve taranmasına kadar neredeyse her şeyi yapabilecek olanaklara sahip olduğu bilinmektedir.

ActiveX kontrolleri betik dilleri kullanılarak ya da doğrudan HTML OBJECT etiketi vasıtasıyla web sayfalarından çağırılabilir. Bir ActiveX kontrol bilgisayara kurulmamışsa, kontrolün elde edilebileceği URLnin belirtilmesi de mümkündür. Bir kontrol bilgisayara bir kere indirildikten sonra yeniden indirilmeye gerek kalmaksızın tekrar tekrar çağırılabilir. Bu durum özellikle birden fazla kullanıcıya hizmet veren bilgisayarlarda sorun yaratabilmektedir.

ActiveX kontrolleri, Microsoft'un sayısal imza yöntemi, Authenticode ile imzalanmış ya da imzasız olabilir. İmza, kontrolün imzalayan tarafından yaratıldığı ve değiştirilmediği konusunda üst düzey bir doğrulama sağlamakla beraber, kodu üretenin iyi niyeti, güvenirliği ve yeterliği konusunda hiç bir garanti sağlamaz. Belirli bir üreticiden gelen belirli bir kodun ne kadar güvenli olduğuna karar vermek tamamen kullanıcının sorumluluğundadır. ActiveX kontrollerinin imzası kalıcıdır. Bir ActiveX kontrolü imzalandıktan sonra saldırganların hedefi haline gelmektedir. İmzalı bir kontrolde açık fark edilirse, saldırganlar kullanıcı ile imzalayan arasındaki güven ilişkisini kullanarak açıktan faydalanmakta ve kullanıcıya zarar verebilmektedir. [2], [3]

5 Sonuç

İnternet kullanımı yaygınlaştıkça, mobil kodun web sayfası tasarımı ve kullanımına getirdiği kolaylıklar nedeniyle mobil kod kullanımı da yaygınlaşmaktadır. Kullanıcı ve üreticiler için ilgi uyandıran her şey saldırganların da ilgi odağı olmaktadır. Bu durumda mobil kod güvenliği sorunu en büyük engel olarak karşımıza çıkmaktadır. Güvenlik ile ilgili çalışmalar devam etmekle beraber, kesin bir çözümün ortaya çıkarıldığı söylenemez.

Mobil kod güvenliği ile ilgili olarak yapılan çalışmalar daha çok ortamın koddan korunmasına yöneliktir. Kodun işletim ortamından korunması ile ilgili tatmin edici sonuçlar bulunmamaktadır.

İnternet üzerinden yapılan işlemler arttıkça, güvenlik yöntemlerinin geliştirilmesi ve çeşitlendirilmesi ihtiyacı da artacaktır. Bu yönde çalışmalar devam etmekte, endüstriyel anlamda çözümler de ortaya çıkmaktadır.

Kaynakça

1. McGraw, G. ve Felten, E., Securing Java, John Wiley And Sons Inc., 1999, ISBN:0-471-31952-X
2. Microsoft Corp, Improving Web Application Security: Threats & Counter Measures, http://msdn.microsoft.com/library/en-us/dnnetsec/html/ThreatCounter.asp 2003.
3. CERT Coordination Center, Results of the Security in ActiveX Workshop, Aralık 2000.
4. Colby, C., Crary, K. Harper, R., Lee, P. ve Pfenning, F., Automated Techniques For Provably Safe Mobile Code, Theoretical Computer Science 290 (2003) s.1175-1199.
5. Oppliger R., Security Issues Related to Mobile Code and Agent Based Systems, Computer Communications 22 (1999) s. 1165-1170.
6. Fritzinger, J.S ve Mueller, M., Java Security Whitepaper, Sun Microsystems Inc, 1996.
7. Banino, J. Parallelism and Fault Tolerance in Chorus. Journal of Systems and Software, 1986 s. 205–211.
8. Johansen, D., van Renesse, R., Schneider, F.B., Operating System Support for Mobile Agents. In: Proceedings of the Fifth IEEE Workshop on Hot Topics in Operating Systems (HotOS-V), Mayıs 1995, s. 42–45.
9. Jul, E., Levy, H., Hutchinson, N., Black, A., Fine-grained Mobility in the Emerald System. ACM Transactions on Computer Systems 6 (1988), s. 109–133.
10. Stamos, J.W., Gifford, D.K., Remote Evaluation. ACM Transactions on Programming Languages and Systems 12 (1990), s. 537–565.
11. Vittal, J., Active Message Processing: Messages as Messengers, Computer Message System. North-Holland Amsterdam, 1981, s. 175–195.
12. White, J.E., Mobile Agents. Teknik Rapor, General Magic, Ekim 1995
13. Thomas, R., A Survey of Mobile Code Security Techniques, 22nd National Information Systems Security Conference,

Ekim 1999.

14. Gray, R.S., Agent Tcl: A Flexible and Secure Mobile-Agent System, Proceedings of the fourth Annual Tcl/Tk Workshop (TCL96), 1996, s. 9-23.

15. Vigna, G., Mobile Code Technologies, Paradigms and Applications, Politecnico Di Milano Doktora Tezi, 1999.