

Java Paket Yapısını Bozarak Kod Gizleme

Erdem GÜVEN, Şen ÇAKIR

Dokuz Eylül Üniversitesi, Bilgisayar Mühendisliği Bölümü, İzmir
erdem.guven@senvion.com, sen@cs.deu.edu.tr

Özet: Java sınıf dosyalarının önemli bir zayıflığı kaynak kod hakkında çok fazla bilgi içermeleridir. Geri derleme araçları yardımı ile bu dosyalardan kolaylıkla kaynak koda ulaşılabilmektedir. Java paket yapısı, kod analizlerini kolaylaştırıcı bilgiler taşımaktadır. Yazılım ürünlerini, tersine mühendislik çalışmalarından koruyabilmek için kod gizleme yöntemleri geliştirilmiştir fakat şu ana kadar yapılan çalışmalardan hiçbiri paket yapısı üzerine eğilmemiştir. Biz bu çalışmamızda paket yapısındaki bilgileri değersiz kılmak için iki yöntem öneriyoruz: paketsizleştirme ve rastgele paketleme. Bu bildiride yöntemlerin uygulaması ve yan etkileri üzerinde durulmuştur.

Anahtar Kelimeler: Kod Gizleme, Tersine Mühendislik, Geri Derleme.

Obfuscation By Altering Java Package Structure

Abstract:Java class files have a major weakness that they include considerable information about source code. It's easy to decompile these files and reconstruct sources. Java package structure obtains valuable information for code analyses. Although many obfuscation techniques have been developed to protect software products from reverse engineering threats, none of them focused on package structure. We propose two techniques: removing packages and random packaging. In this paper applications and side effect of the techniques are being studied.

Keywords: Obfuscation, Reverse Engineering, Decompiling.

1. Giriş

Tersine mühendislik, bir ürünü inceleyerek üretim teknikleri, mimari ve mühendislik bilgilerinin çıkarılmasıdır. Bu çalışma kaybolan bilgileri tekrar oluşturma amaçlı yapılmasının yanında bilgi hırsızlığı için de kullanılmaktadır.

Tersine mühendislik birçok ürün üzerinde uygulanmaktadır. Örneğin, mekanik, elektronik ürünler, kimyasallar, ilaçlar, gıdalar ve yazılım gibi. Yazılım ürünleri mühendislik çalışmalarının çoğunu ürün içinde yalın bir şekilde içerdiği için, diğerlerinden daha hassas bir konumdadır. Geri derleyiciler vasıtası ile yazılım ürünlerinin ana formu olan ikili dosyalardan kaynak kodlar yüksek başarımlarında elde edilebilmektedir.

Java mimarisi son yıllarda çokça popülerleşmiş bir yazılım ortamıdır. Platform bağımsızlığı sayesinde kişisel bilgisayarlarda, sunucularda, gömülü sistemlerde, ağ sayfalarında ve daha birçok yerde kullanılmaktadır.

Java mimarisi Java programlama dilini ve Java sanal makinesini içerir. Java programlama dili modern nesne tabanlı bir yapı üstüne kurulmuştur. Java sanal makinesi de bu programlama diliyle büyük bir uyumluluk içinde tasarlanmıştır. Java dilinin ihtiyaç duyduğu yığıt yönetimi, hafıza yönetimi, çöp toplayıcı, tip kontrolü gibi birçok temel servis, makine tarafından sağlanmaktadır. Java dili ile Java makine kodları arasında da büyük bir paralellik vardır.

Java'nın platform bağımsızlığını ve taşınabilirliğini sağlayan temel özelliklerinden biri dinamik bağlamadır. Bu özelliği sağlamak için sınıf dosyalarının içinde bu bağlarla ilgili birçok bilgi bulunmaktadır. Bir sınıf dosyası kaynak kod hakkında birçok bilgi içermektedir.

İkili kod ile üst seviye kod arasındaki paralellik ve sınıf dosyalarının çok bilgi içermesinden dolayı derlenmiş Java sınıf dosyaları neredeyse kayıpsız olarak kaynak koda dönüştürülebilmektedir.

Yazılımları tersine mühendislik çalışmalarından koruyabilmek için birçok çalışma yapılmıştır. Çabalar, sunucu tarafında işlem, şifreleme, donanımsal kod koruma ve kod gizleme üzerinde yoğunlaşmaktadır. Sunucu tarafında işlemde korunmak istenen kod sunucu üzerinde çalışmaktadır. Kullanıcılar koda ulaşamadıkları için kod korunma altındadır. Sunucu ile sürekli bağlantı gerektirdiğinden bu yöntem her zaman kullanım alanı bulamamaktadır. Şifrelenmiş kod ise çalıştırılmadan önce şifrenin açılması gerektiği için tam bir koruma sağlamaz. Donanımsal kod gizlemede yazılımın kullanıcı tarafından okunması donanım tarafından engellenir. Bu yöntem bütün donanımlarda uygulanmadan başarı bulamaz ve uygulanması sıkıntılıdır. Kod gizleme kodun işleyişini değiştirmeden kodu incelenmesi daha zor hale getirmektir. Bu yöntemde tersine mühendisliği imkânsız hale getirmek mümkün değildir. Yapılmaya çalışılan tersine mühendislik maliyetini yazılımın değerinden daha yükseğe çıkarmak ve tersine mühendislik çalışmasını anlamsız kılmaktır.

Biz bu çalışmamızda Java paketleme yapısı üzerine eğildik. Paketleme yapısı genellikle benzer işler üstlenen sınıfları bir araya getirmek ve diğer sınıflardan bağımsızlaştırmak için kullanılır. Bu gruplama tersine mühendislik için önemli bilgiler içerir. Çalışmamızda

bu bilgileri yararsız kılacak iki yöntem öneriyoruz: paketsizleştirme ve rastgele paketleme.

2. İlgili Çalışmalar

Şu ana kadar kod gizleme üzerine yapılmış birçok çalışma bulunmaktadır. İlk çalışmalar daha çok düşük seviye kodların tekrar düzenlenmesi üzerine oldu. Cohen yaptığı çalışmada program komutlarını ve parçalarını işlem sonucunu değiştirmeyecek şekilde farklılaştırmaya eğildi. Kullandığı teknikler arasında komut sırası değiştirme, rastgele dallanma ekleme ve metot oluşturma vardır [1].

Collberg ve arkadaşları, sözlüksel gizleme (isim değiştirme) ve veri dönüştürme yöntemlerini geliştirdiler [2]. Asıl önemli katkıları ise komut akışı dönüşümleridir. Boş yüklemeler vasıtasıyla ölü kod ekleme yöntemini önerdiler.

Sakabe ve arkadaşları, Java ve nesne tabanlı yapısı üzerine eğildiler. Çok biçimlilik özelliğinden faydalanarak bütün metot dönüş tiplerini bir sınıfta topladılar ve metotlar bu sınıfı dönecek şekilde değiştirildi [3]. Aynı işlemi metot parametreleri üzerinde de uyguladılar. Bunların yanında farklı sınıfların aynı metot isimlerini kullanmasını sağladılar. Böylece metotların gerçek tanımlarının saklandığı gibi aynı tanıma sahip birçok metot da oluşturuldu. Sağlanan iyi kod gizleme, yanında %30 yavaşlama ve %300 kod büyüklüğünde artma getirdi.

Sonsonkin ve arkadaşları, yüksek seviyeli üç kod gizleme yöntemi önerdiler [4]. Bunlar birçok sınıfı tek bir sınıf altında toplama, bir sınıfı birçok sınıfa ayırma ve tip gizlemedir.

Yüksek seviyeli kod gizleme yöntemlerinin hiçbiri Java paketleme yapısı üzerinde durmamıştır. Biz bu yapıyı bozmaya ve tersine mühendislik için faydasız kılmaya çalıştık.

3. Bizim Katkımız

Java paketleri sınıflar arasında yüksek seviyeli organizasyon bilgisi tutar ve tersine mühendislik için önemli bir kaynaktır. Örnek bir paket yapısı Şekil 1’de görülebilir.

```
com
|-- cdsc
|-- eje
|-- entities
|-- gui
|-- options
|-- search
|-- utilities
|-- jdk
|-- exceptions
|-- utilities
|-- utilities
```

Şekil 1. Örnek Paket Yapısı

Kod analizi yapan bir kişi ‘gui’ paketi altındaki sınıfların bir kısmını inceledikten sonra bu paketdeki sınıfların kullanıcı ara birimi ile ilgili olduğunu anlayacaktır (koda en azından ‘isim karıştırma’ kod gizleme yöntemi uygulandığını, paket ve sınıfların anlamsız isimler aldıklarını kabul ediyoruz) ve amacına göre ya bu paketi incelemeyi bırakacak yada daha da yoğunlaşacaktır.

Paket bilgisinin analizlerdeki bu tür kolaylaştırıcı etkilerinden kurtulmak için paketlemenin kullanılmaması (paketsizleştirme) yada paketlenen sınıflar arasında işlevsel benzerliklerin olmaması (rastgele paketleme) gerekmektedir. Kod geliştirimi sırasında bu tür bir yöntem izlemek paketlemenin tasarım, geliştirim ve test süreçlerine getirdiği kolaylıklardan faydalanamamanın yanında fazladan çaba sarf etmeye zorlayacaktır. Bu durumda paket bilgisinden kurtulmanın en ideal yolu paketsizleştirme ve rastgele paketleme yöntemlerinin yardımcı bir uygulama vasıtasıyla geliştirme ve sınama süreçlerinden sonra uygulanmasıdır. Çalışmamız sırasında bu tür yardımcı bir uygulama geliştirerek yöntemlerimizi sınadık.

3.1. Paketsizleştirme

Paketlerin içerdiği bilgidan kurtulmanın en kolay yolu sınıfları tamamen paketlerden çıkarmak ya da birkaç paket altında toplamaktır (Şekil 2). Bu işlem sırasında bu sınıflara olan bütün referansların güncellenmesi gerekmektedir, böylece dinamik bağlama sırasında istenen sınıf bulunabilecektir. Dikkat edilmesi gereken bir nokta paket içi erişim kısıtlaması getirilmiş öğelerdir. Bu öğeler güvenlik ve tasarım kararları doğrultusunda sadece aynı paketdeki sınıfların erişimine açıktır. Paketsizleştirme sonunda bütün sınıflar aynı paket içinde yer aldığı için bu kısıtlamalar işlev dışı duruma gelmektedir.

```
pac1 | npac
|-- pac11 | |-- cls111.class
|-- cls111.class | |-- cls112.class
|-- cls112.class | |-- cls1131.class
|-- pac113 | |-- cls121.class
|-- cls1131.class |
|-- pac12 |
|-- cls121.class |
```

Şekil 2. Paketsizleştirme Örneği

3.2. Rastgele Paketleme

Bu yöntemde paketleme bilgisini yok etmek yerine sınıfları rastgele paketleyerek tersine mühendislik çalışması yapanların akıllarını karıştırmak hedeflenmektedir (Şekil 3). Paketsizleştirme sırasında olduğu gibi bu yöntemde de referansların güncellenmesi gerekmektedir. Yine aynı şekilde erişim kısıtlamalar üzerinde düşünülmesi gereken noktalar vardır. Paketsizleştirmeden farklı olarak paket içi erişim kısıtlamalarının etkinliğini yitirmenin yanında bu kısıtlamaları tamamen kaldırıp öğeleri genel erişime açmak gerekebilmektedir. Bu işlemin sebebi eskiden aynı paket içinde yer alan sınıfların rastgele paketleme sonunda farklı paketlere düşme durumudur. Böyle bir durumda paket içi erişim kısıtlamalarını koruyan öğelere diğer sınıflardan erişim sağlanamayacaktır. Uygulanabilecek tek çözüm öğeleri genel erişime açmaktır.

```
pac1
|-- pac11
| |-- cls111.class
| |-- cls112.class
|   |-- pac113
|   |-- cls1131.class
|-- pac12
|-- cls121.class

npac1
|-- cls111.class
|-- npac11
| |-- npac112
| |-- cls1131.class
|-- npac12
| |-- cls112.class
|-- npac13
|-- cls121.class
```

Şekil 3. Rastgele Paketleme Örneği

4. Uygulama

Yöntemlerimizi sınamak için Java’da basit bir kod gizleme uygulaması geliştirilmiştir. Uygulamamız derleme sonrası ikili formdaki sınıf dosyalarını işlemektedir. İkili dosyaları ayrıştırmak ve işlemek için ASM[5] Java ikili kod işleme ve analiz kütüphanesinden faydalanılmıştır. Uygulamamız örnek Java uygulamaları üzerinde paketsizleştirme ve rastgele paketleme yöntemlerini başarılı olarak uygulamıştır.

5. Sonuç ve Gelecek Çalışmalar

Bu çalışmamızda, iki kod gizleme yöntemi önerilmiş olup, yöntemlerin uygulanması ve yan etkileri üzerinde durulmuştur. Java paket yapısı yazılım hakkında kayda değer bilgiler içermektedir. Bu yapının kaldırılması / değiştirilmesi tersine mühendislik çalışmalarını zorlaştıracaktır.

Sonraki çalışmalarımızda yöntemlerimizin etkinliğini teorik olarak ve deneysel verilerle göstermeye çalışılacaktır. Geliştirdiğimiz uygulama açık kaynak kodlu kod gizleyicilerle birleştirilip daha kullanışlı hale getirilecektir.

6. Kaynaklar

- [1]. Cohen F. B., “Operating system protection through program evolution”, Computers and Security, Cilt: 12, Sayı: 6, 1993, 565 – 584.
- [2]. Collberg C. S. ve Thomborson C., “Watermarking, tamper-proofing, and obfuscation – tools for software protection”, IEEE Transactions on Software Engineering, Cilt: 28, Sayı: 8, 2002, 735–746.
- [3]. Sakabe Y., Soshi M. ve Miyaji A., “Java obfuscation with a theoretical basis for building secure mobile agents”, Communications and Multimedia Security, 2003, 89–103.
- [4]. Sosonkin M., Naumovich G. ve Memon N., “Obfuscation of design intent in object-oriented applications”, DRM ’03: Proceedings of the 3rd ACM workshop on Digital rights management, 142–153, 2003, New York, ABD.
- [5]. ASM – Home Page, asm.objectweb.org, OW2 Consortium.