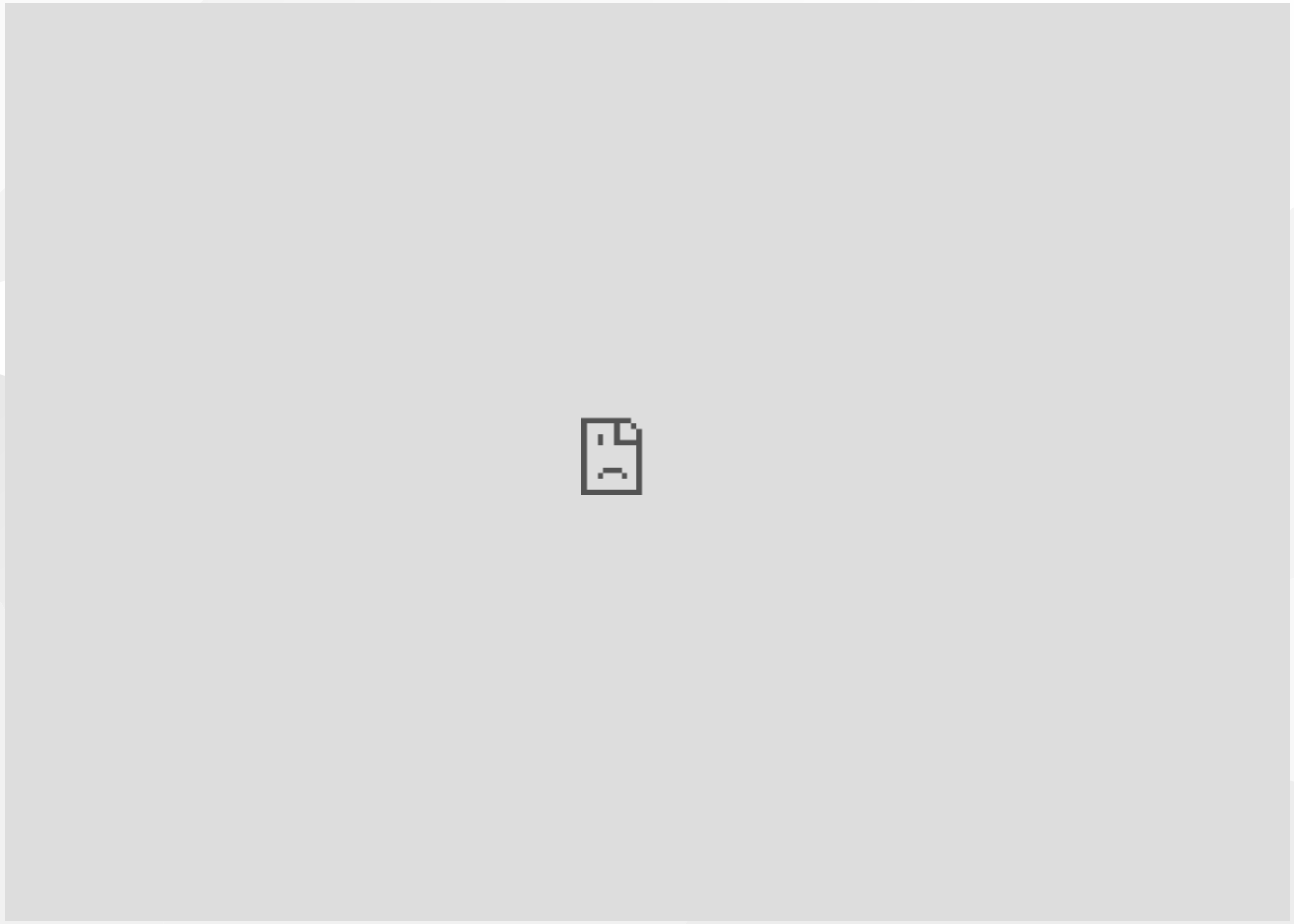
CEN429 Güvenli Programlama

Hafta-4

Kod Güçlendirme Teknikleri

İndir

- PDF
- DOC
- SLIDE
- PPTX



Outline

- Kod Güçlendirme Teknikleri
- Native C/C++ İçin Kod Güçlendirme
- Java ve Yorumlanan Diller İçin Kod Güçlendirme

Hafta-4: Kod Güçlendirme Teknikleri

1. Native C/C++ İçin Kod Güçlendirme Teknikleri

C ve C++ gibi düşük seviye dillerde güvenli kod yazmak ve saldırılara karşı dayanıklı hale getirmek için çeşitli teknikler kullanılır. Bu teknikler, kodun analiz edilmesini ve geri mühendislik işlemlerini zorlaştırmayı amaçlar.

a) Opaque Loops (Opak Döngüler)

Teorik Açıklama: Opak döngüler, dışarıdan bakıldığında amacı belli olmayan döngülerdir. Bu döngüler sayesinde kodun analizi zorlaşır. Saldırgan, döngünün işlevini anlamakta zorlanır ve kodun çözülmesi daha karmaşık hale gelir.

Uygulama Örnekleri:

1. Rastgele bir koşul ile oluşturulmuş döngüler ekleyerek kodun analizini zorlaştırma.
2. Dışarıdan anlaşılmayan ancak programın işleyişine zarar vermeyen döngüler ekleme.
3. Opak döngüler ile programın çalışma süresini arttırarak saldırganı yanıltma.

b) Shared Object Sembollerini Gizleme (Configure Shared Object Symbol Invisible)

Teorik Açıklama: Paylaşılan nesneler (shared object) içinde kullanılan sembollerin gizlenmesi, bu nesnelere dışarıdan erişimi zorlaştırır. Bu işlem, analiz ve geri mühendislik işlemlerini engellemek için kullanılır.

Uygulama Örnekleri:

1. Derleyici seçenekleriyle sembollerin görünürlüğünü sınırlama.
2. Sadece gerekli sembolleri dışa açarak diğer sembollerin erişilemez olmasını sağlama.
3. Paylaşılan kütüphanelerdeki kritik fonksiyonları gizleyerek güvenliği artırma.

c) Aritmetik İşlemlerin Obfuske Edilmesi (Obfuscation of Arithmetic Instructions)

Teorik Açıklama: Aritmetik işlemler, programın en temel yapı taşlarıdır. Bu işlemleri karmaşık hale getirmek, kodun analizini ve anlaşılmasını zorlaştırır.

Uygulama Örnekleri:

1. Basit toplama işlemlerini daha karmaşık matematiksel ifadeler ile değiştirme.
2. Aritmetik işlemlerine gereksiz adımlar ekleyerek işlevselliği korurken kodun anlaşılmasını zorlaştırma.
3. Aritmetik işlemler üzerinde bit manipülasyonu yaparak daha karmaşık hale getirme.

d) Fonksiyon İsimlerinin Obfuske Edilmesi (Obfuscation of Function Names)

Teorik Açıklama: Fonksiyon isimlerinin rastgele karakter dizileri ile değiştirilmesi, kodun anlaşılmasını zorlaştırır. Bu teknik, özellikle tersine mühendislik (reverse engineering) işlemlerini engellemek için kullanılır.

Uygulama Örnekleri:

1. Fonksiyon isimlerini anlamsız karakter dizileri ile değiştirme.
2. Her derlemede farklı fonksiyon isimleri oluşturarak statik analiz araçlarını yanıltma.
3. Kritik fonksiyonların isimlerini rastgele hale getirerek saldırganların bu fonksiyonları anlamasını zorlaştırma.

e) Kaynak Dosya İsimlerinin Obfuske Edilmesi (Obfuscation of Source File Names)

Teorik Açıklama: Kaynak dosyaların isimlerini anlamsız hale getirerek kodun hangi fonksiyona veya sınıfa ait olduğunu gizleme.

Uygulama Örnekleri:

1. Kaynak dosyaların isimlerini rastgele karakterler ile değiştirme.
2. Kaynak dosyalar arasındaki ilişkiyi gizleyerek kod yapısını anlaşılmaz hale getirme.
3. Dosya isimlerini obfuske ederken kaynak kodu etkilemeyecek şekilde yapıları değiştirme.

f) Statik Dizelerin Obfuske Edilmesi (Obfuscation of Static Strings)

Teorik Açıklama: Statik dizeler, saldırganlar için önemli bilgi kaynaklarıdır. Bu dizelerin şifrelenmesi ve gizlenmesi, kod güvenliğini artırır.

Uygulama Örnekleri:

1. Statik dizeleri şifreleyerek çalışma anında çözülmesini sağlama.
2. Rastgele dize maskeleri uygulayarak dizelerin anlamını gizleme.
3. Dize sabitlerini kaldırarak sabit dize kullanımını azaltma.

g) Diğer Kod Güçlendirme Teknikleri

1. **Opaque Boolean Variables:** Koşullu ifadelerin karmaşık hale getirilmesi.
2. **Function Boolean Return Codes:** Fonksiyon dönüş değerlerinin karmaşıklaştırılması.
3. **Obfuscation of Function Parameters:** Fonksiyon parametrelerinin gizlenmesi.
4. **Bogus Function Parameters & Operations:** Anlamsız parametreler ve işlemler ekleyerek kodun analizini zorlaştırma.
5. **Control Flow Flattening:** Kontrol akışını düzleştirerek tahmin edilemez hale getirme.
6. **Randomized Exit Points:** Çıkış noktalarını rastgele hale getirerek kodun öngörülebilirliğini azaltma.
7. **Logging Disabled on Release:** Son sürümde loglamaların devre dışı bırakılması.

2. Java ve Yorumlanan Diller İçin Kod Güçlendirme Teknikleri

Java ve diğer yorumlanan dillerde kod güçlendirme, güvenlik açıklarını azaltmak ve geri mühendislik işlemlerini zorlaştırmak için kullanılır.

a) Proguard ile Kod Obfuske ve Koruma (Proguard Code Obfuscation and Code Shrink Protection)

Teorik Açıklama: Proguard, Java kodlarını küçültme, optimize etme ve obfuske ederek kodun analiz edilmesini zorlaştırır.

Uygulama Örnekleri:

1. Proguard yapılandırma dosyası ile kodun küçültülmesi ve optimize edilmesi.
2. Obfuske edilmiş kodun test edilmesi ve hataların çözülmesi.
3. Proguard raporlarının analizi ile hangi öğelerin obfuske edildiğinin tespiti.

b) Cihaz Bağlama İçin Ayrı Parmak İzi Depolama (Separated Fingerprint Storage for Device Binding)

Teorik Açıklama: Cihazın benzersiz özelliklerini kullanarak, uygulamanın yalnızca belirli bir cihazda çalışmasını sağlamak için kullanılan bir tekniktir.

Uygulama Örnekleri:

1. Cihaz parmak izinin şifrelenerek güvenli bir şekilde depolanması.
2. Parmak izi doğrulaması ile uygulamanın cihaz üzerinde çalışmasını sağlama.
3. Parmak izi verilerinin gizlenmesi ve saldırılara karşı korunması.

c) Yerel Kütüphane JNI API Obfuske Etme (Native Library JNI API Obfuscation)

Teorik Açıklama: Java Native Interface (JNI) kullanılarak çağrılan yerel kütüphanelerin obfuske edilmesi, geri mühendislik işlemlerini zorlaştırır.

Uygulama Örnekleri:

1. JNI fonksiyon isimlerinin rastgele karakterlerle değiştirilmesi.
2. JNI parametrelerinin gizlenmesi ve anlaşılmasını zorlaştırma.
3. JNI hata yönetimi ile saldırganların hataları analiz etmesini engelleme.

d) Statik Dizelerin Obfuske Edilmesi (Obfuscation of Static Strings)

Teorik Açıklama: Statik dizeler, saldırganların geri mühendislik işlemleri sırasında kullanabileceği önemli bilgiler içerir. Bu dizelerin obfuske edilmesi, güvenliği artırır.

Uygulama Örnekleri:

1. Statik dizelerin şifrelenmesi ve çalışma anında çözülmesi.
2. Dizelerin obfuske edilerek anlamlarının gizlenmesi.
3. Rastgele dize oluşturma ve manipülasyon teknikleri ile güvenliği artırma.

Haftanın Özeti ve Gelecek Hafta

Bu Hafta:

- Kod Güçlendirme Teknikleri (C/C++ ve Java)
- Obfuske Teknikleri ve Uygulamaları

Gelecek Hafta:

- Saldırı Ağaçları ve Güvenlik Modelleri
- Saldırı Yöntemleri ve Güvenli İletişim

$4. Hafta - Sonu$