

1.2.0.2 a) Opaque Loops (Opak Döngüler) Teorik Açıklama: Opak döngüler, döngüden bakıldığında amaç belli olmayan döngülerdir. Bu döngüler sayesinde kodun analizi zorlaştır. Saldırgan, döngünün işlevini anlamakta zorlanır ve kodun işleyişi daha karmaşık hale gelir.

Uygulama Örnekleri:

1. Rastgele bir koşul ile oluşturulmuş döngüler ekleyerek kodun analizini zorlaştırma.
2. Döngüden anlaşılmayan ancak programın işlevine zarar vermeyen döngüler ekleme.
3. Opak döngüler ile programın işlevini artırarak saldırıya karşı direnç artırma.

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

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

Uygulama Örnekleri:

1. Derleyici seçenekleriyle sembollerin görünürlüğünü artırma.
2. Sadece gerekli sembollerin tanımlanmasıyla sembollerin erişilemez olması sağlanabilir.
3. Paylaşılan kütüphanelerdeki kritik fonksiyonların gizlenmesiyle direnç artırma.

1.2.0.4 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 değiştirmek.
2. Aritmetik işlemlerine gereksiz adımlar ekleyerek işlevselliği korurken kodun anlaşılmasını zorlaştırma.
3. Aritmetik işlemler üzerinde bit manipülasyonu yaparak daha karmaşık hale getirme.

1.2.0.5 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 değiştirmek.
2. Her derlemede farklı fonksiyon isimleri oluşturularak statik analiz araçları için direnç artırma.
3. Kritik fonksiyonların isimlerini rastgele hale getirerek saldırırganların bu fonksiyonları anlamasını zorlaştırma.

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 değiştirmek.
2. Kaynak dosyaları arasındaki ilişkileri gizleyerek kod yapısını anlamaz hale getirme.
3. Dosya isimlerini obfuske ederken kaynak kodu etkilemeyecek şekilde yapıları değiştirmek.

1.2.0.6 f) Statik Dizilerin Obfuske Edilmesi (Obfuscation of Static Strings) Teorik Açıklama: Statik diziler, saldırırganların önemli bilgi kaynaklarıdır. Bu dizilerin şifrelenmesi ve gizlenmesi, kodun erişimini artırır.

Uygulama Örnekleri:

1. Statik dizileri şifreleyerek işlevi artırma ve işleyişini sağlaması.

2. Rastgele dize maskeleri uygulayarak dizelerin anlamÄ±nÄ± gizleme.
3. Dize sabitlerini kaldÄ±rarak sabit dize kullanÄ±mÄ±nÄ± azaltma.

1.2.0.7 g) DiÄŸer Kod GÄ¼ÄŸlendirme Teknikleri

1. **Opaque Boolean Variables:** KoÄŸullu ifadelerin karmaÄŸÄ±k hale getirilmesi.
2. **Function Boolean Return Codes:** Fonksiyon dÄŸerlerinin karmaÄŸÄ±klaÄŸtÄ±rÄ±lmasÄ±.
3. **Obfuscation of Function Parameters:** Fonksiyon parametrelerinin gizlenmesi.
4. **Bogus Function Parameters & Operations:** AnlamsÄ±z parametreler ve iÄŸlemler ekleyerek kodun analizini zorlaÄŸtÄ±rma.
5. **Control Flow Flattening:** Kontrol akÄ±ÄŸÄ±nÄ± dÄŸizleyerek tahmin edilemez hale getirme.
6. **Randomized Exit Points:** Ä±Ä±kÄ± noktalarÄ±nÄ± rastgele hale getirerek kodun ÄŸngÄŸrÄ¼lebilirliÄŸini azaltma.
7. **Logging Disabled on Release:** Son sÄ¼rÄ¼mde loglamalarÄ±n devre dÄ±ÄŸÄ± bÄ±rakÄ±lmasÄ±.

1.2.1 2. Java ve Yorumlanan Diller Ä°ÄŸin Kod GÄ¼ÄŸlendirme Teknikleri

Java ve diÄŸer yorumlanan dillerde kod gÄ¼ÄŸlendirme, gÄ¼venlik aÄŸÄ±klarÄ±nÄ± azaltmak ve geri mÄ¼hendislik iÄŸlemlerini zorlaÄŸtÄ±rmak iÄŸin kullanÄ±lÄ±r.

1.2.1.1 a) Proguard ile Kod Obfuske ve Koruma (Proguard Code Obfuscation and Code Shrink Protection) Teorik AÄŸÄ±klama: Proguard, Java kodlarÄ±nÄ± kÄ¼ÄŸÄ¼ltme, optimize etme ve obfuske ederek kodun analiz edilmesini zorlaÄŸtÄ±rÄ±r.

Uygulama Ä±rneklere:

1. Proguard yapÄ±landÄ±rma dosyasÄ± ile kodun kÄ¼ÄŸÄ¼ltÄ±lmesi ve optimize edilmesi.
2. Obfuske edilmiÄŸ kodun test edilmesi ve hatalarÄ±n ÄŸzÄ¼lmesi.
3. Proguard raporlarÄ±nÄ±n analizi ile hangi ÄŸÄŸelerin obfuske edildiÄŸinin tespiti.

1.2.1.2 b) Cihaz BaÄŸlama Ä°ÄŸin AyrÄ± Parmak Ä°zi Depolama (Separated Fingerprint Storage for Device Binding) Teorik AÄŸÄ±klama: CihazÄ±n benzersiz ÄŸzelliklerini kullanarak, uygulamanÄ±n yalnÄ±zca belirli bir cihazda ÄŸsalÄ±ÄŸmasÄ±nÄ± saÄŸlamak iÄŸin kullanÄ±lan bir tekniktir.

Uygulama Ä±rneklere:

1. Cihaz parmak izinin ÄŸifrelenerek gÄ¼venli bir ÄŸekilde depolanmasÄ±.
2. Parmak izi doÄŸrulamasÄ± ile uygulamanÄ±n cihaz Ä¼zerinde ÄŸsalÄ±ÄŸmasÄ±nÄ± saÄŸlama.
3. Parmak izi verilerinin gizlenmesi ve saldÄ±rÄ±lara karÄŸÄ± korunmasÄ±.

1.2.1.3 c) Yerel KÄ¼tÄ¼phane JNI API Obfuske Etme (Native Library JNI API Obfuscation) Teorik AÄŸÄ±klama: Java Native Interface (JNI) kullanÄ±larak ÄŸaÄŸrÄ±lan yerel kÄ¼tÄ¼phanelerin obfuske edilmesi, geri mÄ¼hendislik iÄŸlemlerini zorlaÄŸtÄ±rÄ±r.

Uygulama Ä±rneklere:

1. JNI fonksiyon isimlerinin rastgele karakterlerle deÄŸiÄŸtirilmesi.
2. JNI parametrelerinin gizlenmesi ve anlaÄŸÄ±lmasÄ±nÄ± zorlaÄŸtÄ±rma.
3. JNI hata yÄŸnetimi ile saldÄ±rganlarÄ±n hatalarÄ± analiz etmesini engelleme.

1.2.1.4 d) Statik Dizelerin Obfuske Edilmesi (Obfuscation of Static Strings) Teorik AÄŸÄ±klama: Statik dizeler, saldÄ±rganlarÄ±n geri mÄ¼hendislik iÄŸlemleri sÄ±rasÄ±nda kullanabileceÄŸi ÄŸnemli bilgiler iÄŸerir. Bu dizelerin obfuske edilmesi, gÄ¼venliÄŸi artÄ±rÄ±r.

Uygulama Ä±rneklere:

1. Statik dizelerin ÄŸifrelenmesi ve ÄŸsalÄ±ÄŸma anÄ±nda ÄŸzÄ¼lmesi.
2. Dizelerin obfuske edilerek anlamlarÄ±nÄ± gizlenmesi.
3. Rastgele dize oluÄŸturma ve manipÄ¼lasyon teknikleri ile gÄ¼venliÄŸi artÄ±rma.

1.3 Haftanın Özeti ve Gelecek Hafta

1.3.1 Bu Hafta:

- Kod Gözetim ve Kontrol Teknikleri (C/C++ ve Java)
- Obfuscation Teknikleri ve Uygulamaları

1.3.2 Gelecek Hafta:

- Saldırı ve Savunma Teknikleri ve Güvenlik Modelleri
- Saldırı ve Savunma Yöntemleri ve Güvenli İletişim

4. Hafta – Sonu