CEN429 GÃ¼venli Programlama Hafta-4

Kod GÃ¼Ã§lendirme Teknikleri

Yazar: Dr. Ã–ÄŸr. Ãœyesi UÄŸur CORUH

# CEN429 GÃ¼venli Programlama

## Hafta-4

#### Kod GÃ¼Ã§lendirme Teknikleri

Ä°ndir

* [PDF](pandoc_cen429-week-4.pdf)
* [DOC](pandoc_cen429-week-4.docx)
* [SLIDE](cen429-week-4.pdf)
* [PPTX](cen429-week-4.pptx)

### Outline

* Kod GÃ¼Ã§lendirme Teknikleri
* Native C/C++ Ä°Ã§in Kod GÃ¼Ã§lendirme
* Java ve Yorumlanan Diller Ä°Ã§in Kod GÃ¼Ã§lendirme

## **Hafta-4: Kod GÃ¼Ã§lendirme Teknikleri**

#### **1. Native C/C++ Ä°Ã§in Kod GÃ¼Ã§lendirme Teknikleri**

C ve C++ gibi dÃ¼ÅŸÃ¼k seviye dillerde gÃ¼venli kod yazmak ve saldÄ±rÄ±lara karÅŸÄ± dayanÄ±klÄ± hale getirmek iÃ§in Ã§eÅŸitli teknikler kullanÄ±lÄ±r. Bu teknikler, kodun analiz edilmesini ve geri mÃ¼hendislik iÅŸlemlerini zorlaÅŸtÄ±rmayÄ± amaÃ§lar.

#### **a) Opaque Loops (Opak DÃ¶ngÃ¼ler)**

**Teorik AÃ§Ä±klama:** Opak dÃ¶ngÃ¼ler, dÄ±ÅŸarÄ±dan bakÄ±ldÄ±ÄŸÄ±nda amacÄ± belli olmayan dÃ¶ngÃ¼lerdir. Bu dÃ¶ngÃ¼ler sayesinde kodun analizi zorlaÅŸÄ±r. SaldÄ±rgan, dÃ¶ngÃ¼nÃ¼n iÅŸlevini anlamakta zorlanÄ±r ve kodun Ã§Ã¶zÃ¼lmesi daha karmaÅŸÄ±k hale gelir.

**Uygulama Ã–rnekleri:**

1. Rastgele bir koÅŸul ile oluÅŸturulmuÅŸ dÃ¶ngÃ¼ler ekleyerek kodun analizini zorlaÅŸtÄ±rma.
2. DÄ±ÅŸarÄ±dan anlaÅŸÄ±lmayan ancak programÄ±n iÅŸleyiÅŸine zarar vermeyen dÃ¶ngÃ¼ler ekleme.
3. Opak dÃ¶ngÃ¼ler ile programÄ±n Ã§alÄ±ÅŸma sÃ¼resini arttÄ±rarak saldÄ±rganÄ± yanÄ±ltma.

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

**Teorik AÃ§Ä±klama:** PaylaÅŸÄ±lan nesneler (shared object) iÃ§inde kullanÄ±lan sembollerin gizlenmesi, bu nesnelere dÄ±ÅŸarÄ±dan eriÅŸimi zorlaÅŸtÄ±rÄ±r. Bu iÅŸlem, analiz ve geri mÃ¼hendislik iÅŸlemlerini engellemek iÃ§in kullanÄ±lÄ±r.

**Uygulama Ã–rnekleri:**

1. Derleyici seÃ§enekleriyle sembollerin gÃ¶rÃ¼nÃ¼rlÃ¼ÄŸÃ¼nÃ¼ sÄ±nÄ±rlama.
2. Sadece gerekli sembolleri dÄ±ÅŸa aÃ§arak diÄŸer sembollerin eriÅŸilemez olmasÄ±nÄ± saÄŸlama.
3. PaylaÅŸÄ±lan kÃ¼tÃ¼phanelerdeki kritik fonksiyonlarÄ± gizleyerek gÃ¼venliÄŸi artÄ±rma.

#### **c) Aritmetik Ä°ÅŸlemlerin Obfuske Edilmesi (Obfuscation of Arithmetic Instructions)**

**Teorik AÃ§Ä±klama:** Aritmetik iÅŸlemler, programÄ±n en temel yapÄ± taÅŸlarÄ±dÄ±r. Bu iÅŸlemleri karmaÅŸÄ±k hale getirmek, kodun analizini ve anlaÅŸÄ±lmasÄ±nÄ± zorlaÅŸtÄ±rÄ±r.

**Uygulama Ã–rnekleri:**

1. Basit toplama iÅŸlemlerini daha karmaÅŸÄ±k matematiksel ifadeler ile deÄŸiÅŸtirme.
2. Aritmetik iÅŸlemlerine gereksiz adÄ±mlar ekleyerek iÅŸlevselliÄŸi korurken kodun anlaÅŸÄ±lmasÄ±nÄ± zorlaÅŸtÄ±rma.
3. Aritmetik iÅŸlemler Ã¼zerinde bit manipÃ¼lasyonu yaparak daha karmaÅŸÄ±k hale getirme.

#### **d) Fonksiyon Ä°simlerinin Obfuske Edilmesi (Obfuscation of Function Names)**

**Teorik AÃ§Ä±klama:** Fonksiyon isimlerinin rastgele karakter dizileri ile deÄŸiÅŸtirilmesi, kodun anlaÅŸÄ±lmasÄ±nÄ± zorlaÅŸtÄ±rÄ±r. Bu teknik, Ã¶zellikle tersine mÃ¼hendislik (reverse engineering) iÅŸlemlerini engellemek iÃ§in kullanÄ±lÄ±r.

**Uygulama Ã–rnekleri:**

1. Fonksiyon isimlerini anlamsÄ±z karakter dizileri ile deÄŸiÅŸtirme.
2. Her derlemede farklÄ± fonksiyon isimleri oluÅŸturarak statik analiz araÃ§larÄ±nÄ± yanÄ±ltma.
3. Kritik fonksiyonlarÄ±n isimlerini rastgele hale getirerek saldÄ±rganlarÄ±n bu fonksiyonlarÄ± anlamasÄ±nÄ± zorlaÅŸtÄ±rma.

**Teorik AÃ§Ä±klama:** Kaynak dosyalarÄ±n isimlerini anlamsÄ±z hale getirerek kodun hangi fonksiyona veya sÄ±nÄ±fa ait olduÄŸunu gizleme.

**Uygulama Ã–rnekleri:**

1. Kaynak dosyalarÄ±n isimlerini rastgele karakterler ile deÄŸiÅŸtirme.
2. Kaynak dosyalar arasÄ±ndaki iliÅŸkiyi gizleyerek kod yapÄ±sÄ±nÄ± anlaÅŸÄ±lmaz hale getirme.
3. Dosya isimlerini obfuske ederken kaynak kodu etkilemeyecek ÅŸekilde yapÄ±larÄ± deÄŸiÅŸtirme.

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

**Teorik AÃ§Ä±klama:** Statik dizeler, saldÄ±rganlar iÃ§in Ã¶nemli bilgi kaynaklarÄ±dÄ±r. Bu dizelerin ÅŸifrelenmesi ve gizlenmesi, kod gÃ¼venliÄŸini artÄ±rÄ±r.

**Uygulama Ã–rnekleri:**

1. Statik dizeleri ÅŸifreleyerek Ã§alÄ±ÅŸma anÄ±nda Ã§Ã¶zÃ¼lmesini saÄŸlama.
2. Rastgele dize maskeleri uygulayarak dizelerin anlamÄ±nÄ± gizleme.
3. Dize sabitlerini kaldÄ±rarak sabit dize kullanÄ±mÄ±nÄ± azaltma.

#### **g) DiÄŸer Kod GÃ¼Ã§lendirme Teknikleri**

1. **Opaque Boolean Variables:** KoÅŸullu ifadelerin karmaÅŸÄ±k hale getirilmesi.
2. **Function Boolean Return Codes:** Fonksiyon dÃ¶nÃ¼ÅŸ deÄŸerlerinin karmaÅŸÄ±klaÅŸtÄ±rÄ±lmasÄ±.
3. **Obfuscation of Function Parameters:** Fonksiyon parametrelerinin gizlenmesi.
4. **Bogus Function Parameters & Operations:** AnlamsÄ±z parametreler ve iÅŸlemler ekleyerek kodun analizini zorlaÅŸtÄ±rma.
5. **Control Flow Flattening:** Kontrol akÄ±ÅŸÄ±nÄ± dÃ¼zleÅŸtirerek tahmin edilemez hale getirme.
6. **Randomized Exit Points:** Ã‡Ä±kÄ±ÅŸ noktalarÄ±nÄ± rastgele hale getirerek kodun Ã¶ngÃ¶rÃ¼lebilirliÄŸini azaltma.
7. **Logging Disabled on Release:** Son sÃ¼rÃ¼mde loglamalarÄ±n devre dÄ±ÅŸÄ± bÄ±rakÄ±lmasÄ±.

### **2. Java ve Yorumlanan Diller Ä°Ã§in Kod GÃ¼Ã§lendirme Teknikleri**

Java ve diÄŸer yorumlanan dillerde kod gÃ¼Ã§lendirme, gÃ¼venlik aÃ§Ä±klarÄ±nÄ± azaltmak ve geri mÃ¼hendislik iÅŸlemlerini zorlaÅŸtÄ±rmak iÃ§in kullanÄ±lÄ±r.

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

**Teorik AÃ§Ä±klama:** Proguard, Java kodlarÄ±nÄ± kÃ¼Ã§Ã¼ltme, optimize etme ve obfuske ederek kodun analiz edilmesini zorlaÅŸtÄ±rÄ±r.

**Uygulama Ã–rnekleri:**

1. Proguard yapÄ±landÄ±rma dosyasÄ± ile kodun kÃ¼Ã§Ã¼ltÃ¼lmesi ve optimize edilmesi.
2. Obfuske edilmiÅŸ kodun test edilmesi ve hatalarÄ±n Ã§Ã¶zÃ¼lmesi.
3. Proguard raporlarÄ±nÄ±n analizi ile hangi Ã¶ÄŸelerin obfuske edildiÄŸinin tespiti.

#### **b) Cihaz BaÄŸlama Ä°Ã§in AyrÄ± Parmak Ä°zi Depolama (Separated Fingerprint Storage for Device Binding)**

**Teorik AÃ§Ä±klama:** CihazÄ±n benzersiz Ã¶zelliklerini kullanarak, uygulamanÄ±n yalnÄ±zca belirli bir cihazda Ã§alÄ±ÅŸmasÄ±nÄ± saÄŸlamak iÃ§in kullanÄ±lan bir tekniktir.

**Uygulama Ã–rnekleri:**

1. Cihaz parmak izinin ÅŸifrelenerek gÃ¼venli bir ÅŸekilde depolanmasÄ±.
2. Parmak izi doÄŸrulamasÄ± ile uygulamanÄ±n cihaz Ã¼zerinde Ã§alÄ±ÅŸmasÄ±nÄ± saÄŸlama.
3. Parmak izi verilerinin gizlenmesi ve saldÄ±rÄ±lara karÅŸÄ± korunmasÄ±.

#### **c) Yerel KÃ¼tÃ¼phane JNI API Obfuske Etme (Native Library JNI API Obfuscation)**

**Teorik AÃ§Ä±klama:** Java Native Interface (JNI) kullanÄ±larak Ã§aÄŸrÄ±lan yerel kÃ¼tÃ¼phanelerin obfuske edilmesi, geri mÃ¼hendislik iÅŸlemlerini zorlaÅŸtÄ±rÄ±r.

**Uygulama Ã–rnekleri:**

1. JNI fonksiyon isimlerinin rastgele karakterlerle deÄŸiÅŸtirilmesi.
2. JNI parametrelerinin gizlenmesi ve anlaÅŸÄ±lmasÄ±nÄ± zorlaÅŸtÄ±rma.
3. JNI hata yÃ¶netimi ile saldÄ±rganlarÄ±n hatalarÄ± analiz etmesini engelleme.

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

**Teorik AÃ§Ä±klama:** Statik dizeler, saldÄ±rganlarÄ±n geri mÃ¼hendislik iÅŸlemleri sÄ±rasÄ±nda kullanabileceÄŸi Ã¶nemli bilgiler iÃ§erir. Bu dizelerin obfuske edilmesi, gÃ¼venliÄŸi artÄ±rÄ±r.

**Uygulama Ã–rnekleri:**

1. Statik dizelerin ÅŸifrelenmesi ve Ã§alÄ±ÅŸma anÄ±nda Ã§Ã¶zÃ¼lmesi.
2. Dizelerin obfuske edilerek anlamlarÄ±nÄ±n gizlenmesi.
3. Rastgele dize oluÅŸturma ve manipÃ¼lasyon teknikleri ile gÃ¼venliÄŸi artÄ±rma.

## **HaftanÄ±n Ã–zeti ve Gelecek Hafta**

### Bu Hafta:

* **Kod GÃ¼Ã§lendirme Teknikleri (C/C++ ve Java)**
* **Obfuske Teknikleri ve UygulamalarÄ±**

### Gelecek Hafta:

* **SaldÄ±rÄ± AÄŸaÃ§larÄ± ve GÃ¼venlik Modelleri**
* **SaldÄ±rÄ± YÃ¶ntemleri ve GÃ¼venli Ä°letiÅŸim**