CEN429 GÃ¼venli Programlama Hafta-7

Kod Karartma ve Ã‡eÅŸitlendirme Teknikleri

Yazar: Dr. Ã–Ğr. Üyesi UÄŸur CORUH

İçindekiler

Şekil Listesi

Tablo Listesi

1 CEN429 GÃ¼venli Programlama

1.1 Hafta-7

1.1.0.1 Kod Karartma (Obfuscation) ve Ã‡eÅŸitlendirme Teknikleri Ä°ndir

- PDF[1]
- DOC[2]
- SLIDE[3]
- PPTX[4]

1.1.1 Outline

- Kod Karartma ve Ã‡eÅŸitlendirme Teknikleri
- Statik ve Dinamik Kod Karartma
- SanallaÅŸtÄ±rma ve Åžifreleme

1.2 Hafta-7: Kod Karartma (Code Obfuscation) ve Ã‡eÅŸitlendirme (Diversifications)

Kod karartma ve Ã§eÅŸitlendirme teknikleri, yazÄ±lÄ±mÄ±n gÃ¼venliÄŸini artÄ±rmak amacÄ±yla kaynak kodunun ve iÅŸlevlerinin karmaÅŸÄ±k hale getirilmesini iÃ§erir. Bu hafta, bu teknikleri ve bunlarÄ±n uygulamalarÄ±nÄ± inceleyeceÄŸiz. Bu yÃ¶ntemler, Ã¶zellikle yazÄ±lÄ±mlarÄ±n tersine mÃ¼hendislikten korunmasÄ± ve saldÄ±rÄ±larÄ±n zorlaÅŸtÄ±rÄ±lmasÄ± iÃ§in kritik Ã¶neme sahiptir.

[1] pandoc_cen429-week-7.tr_doc.pdf
[2] pandoc_cen429-week-7.tr_word.docx
[3] cen429-week-7.tr_slide.pdf
[4] cen429-week-7.tr_slide.pptx

1.2.0.1 1. Tigress Nedir? Teorik AÃ§Ä±klama: Tigress, programlarÄ± dÃ¶nÃ¼ÅŸtÃ¼rmek, karartmak ve karmaÅŸÄ±k hale getirmek iÃ§in kullanÄ±lan bir araÃ§tÄ±r. Karartma teknikleri ile yazÄ±lÄ±mlarÄ±n tersine mÃ¼hendislikten korunmasÄ±nÄ± saÄŸlar. FarklÄ± karartma teknikleri sunarak kodun analizini zorlaÅŸtÄ±rÄ±r.

1.2.0.2 2. Kod Karartma Teknikleri (Types of Obfuscation) Teorik AÃ§Ä±klama: Kod karartma, kodu insan ve araÃ§lar tarafÄ±ndan anlaÅŸÄ±lmasÄ± zor hale getirir. AÅŸaÄŸÄ±daki teknikler kod karartmanÄ±n temel yÃ¶ntemlerindendir:

- **Abstraction Transformations:** ModÃ¼l yapÄ±larÄ±, sÄ±nÄ±flar, fonksiyonlar vb. yapÄ±larÄ±n yok edilmesi.
- **Data Transformations:** Veri yapÄ±larÄ±nÄ± yeni temsillerle deÄŸiÅŸtirmek.
- **Control Transformations:** Kontrol yapÄ±larÄ±nÄ±n (if, while, repeat vb.) yok edilmesi.
- **Dynamic Transformations:** ProgramÄ±n Ã§alÄ±ÅŸma zamanÄ±nda deÄŸiÅŸiklik yapmasÄ±.

1.2.0.3 3. Statik Kod Karartma (Static Obfuscation) Teorik AÃ§Ä±klama: Statik karartma, programÄ±n Ã§alÄ±ÅŸma zamanÄ±nda sabit kalan karartma tÃ¼rÃ¼dÃ¼r. ProgramÄ±n yapÄ±sÄ±nÄ± deÄŸiÅŸtirir ancak Ã§alÄ±ÅŸÄ±rken deÄŸiÅŸmez. AÅŸaÄŸÄ±daki teknikler bu kategoridedir:

- **Bogus Control Flow:** ProgramÄ±n kontrol akÄ±ÅŸÄ±nÄ± karmaÅŸÄ±k hale getirir. GerÃ§ek olmayan kontrol yapÄ±larÄ± eklenir, Ã¶lÃ¼ dallar ve gereksiz dallar kullanÄ±lÄ±r.
- **Control Flow Flattening:** Kontrol yapÄ±larÄ±nÄ±n yapÄ±larÄ±nÄ± bozarak kodu dÃ¼mdÃ¼z hale getirir.

Uygulama Ã–rnekleri:

1. Kodda gereksiz dallanmalar ve Ã¶lÃ¼ dallar ekleyerek kontrol akÄ±ÅŸÄ±nÄ± zorlaÅŸtÄ±rmak.
2. FonksiyonlarÄ±n iÃ§ine sahte iÅŸlemler yerleÅŸtirmek.

1.2.0.4 4. Opaque Predicates ve KÄ±rma (Breaking Opaque Predicates) Teorik AÃ§Ä±klama: Opaque Predicates, her zaman sabit bir deÄŸere sahip olan, ancak dÄ±ÅŸarÄ±dan bakÄ±ldÄ±ÄŸÄ±nda deÄŸiÅŸiyormuÅŸ gibi gÃ¶rÃ¼nen koÅŸul ifadeleridir. Bu koÅŸullarÄ±n karmaÅŸÄ±k matematiksel veya mantÄ±ksal iliÅŸkilerle oluÅŸturulmasÄ±, kodun analiz edilmesini zorlaÅŸtÄ±rÄ±r.

Uygulama Ã–rnekleri:

1. **Opaque Predicates** kullanarak sabit koÅŸullar oluÅŸturma.
2. Opaque predicatesâ€™i kÄ±rma teknikleri ile matematiksel analizler yaparak bu yapÄ±larÄ± Ã§Ã¶zme.

1.2.0.5 5. Åžifreleme TabanlÄ± SayÄ±sal DÃ¶nÃ¼ÅŸÃ¼mler (Encoding Integer Arithmetic) Teorik AÃ§Ä±klama: SayÄ±lar Ã¼zerinde karmaÅŸÄ±k matematiksel dÃ¶nÃ¼ÅŸÃ¼mler kullanarak orijinal iÅŸlemleri gizleme. Ã–rneÄŸin, toplama iÅŸlemini karmaÅŸÄ±k matematiksel ifadelerle deÄŸiÅŸtirme, tersine mÃ¼hendisliÄŸi zorlaÅŸtÄ±rÄ±r.

Uygulama Ã–rnekleri:

1. **x + y** gibi basit aritmetik iÅŸlemleri gizleyerek yerine daha karmaÅŸÄ±k matematiksel iÅŸlemler yerleÅŸtirme.
2. DÃ¶nÃ¼ÅŸtÃ¼rÃ¼lmÃ¼ÅŸ sayÄ±sal iÅŸlemler Ã¼zerinde Ã§alÄ±ÅŸarak orijinal aritmetik yapÄ±yÄ± geri Ã§Ã¶zme.

1.2.0.6 6. Linear Transformation ve SayÄ±sal DÃ¶nÃ¼ÅŸÃ¼mler (Linear Transformation and Number-Theoretic Tricks) Teorik AÃ§Ä±klama: DoÄŸrusal dÃ¶nÃ¼ÅŸÃ¼mler, orijinal veriyi karmaÅŸÄ±k matematiksel dÃ¶nÃ¼ÅŸÃ¼mlerden geÃ§irerek gizler. Bu dÃ¶nÃ¼ÅŸÃ¼mler geri dÃ¶ndÃ¼rÃ¼lemez deÄŸildir, ancak analiz edilmesi zordur.

Uygulama Ã–rnekleri:

1. Mod 2^32 gibi bÃ¼yÃ¼k modÃ¼ler aritmetiklerle doÄŸrusal dÃ¶nÃ¼ÅŸÃ¼mler yaparak sayÄ±sal iÅŸlemleri gizleme.
2. Euclidâ€™in GeniÅŸletilmiÅŸ Algoritmasİ± gibi matematiksel yÃ¶ntemlerle ters dÃ¶nÃ¼ÅŸÃ¼mleri yapma.

1.2.0.7 7. SanallaÅŸtÄ±rma (Virtualization) Teorik AÃ§Ä±klama: SanallaÅŸtÄ±rma, kodun doÄŸrudan CPU'da Ã§alÄ±ÅŸtÄ±rÄ±lmasÄ± yerine bir sanal makine (interpreter) Ã¼zerinde Ã§alÄ±ÅŸtÄ±rÄ±lmasÄ±nÄ± saÄŸlar. Bu yÃ¶ntemle, programÄ±n Ã§alÄ±ÅŸma zamanÄ±nda sÃ¼rekli olarak Ã§evrimi yapÄ±lÄ±r ve kodun tersine mÃ¼hendisliÄŸi zorlaÅŸtÄ±rÄ±lÄ±r.

Uygulama Ã–rnekleri:

1. ProgramÄ±n tÃ¼m komutlarÄ±nÄ± bir interpreter aracÄ±lÄ±ÄŸÄ±yla Ã§alÄ±ÅŸtÄ±rarak orijinal kodu gizlemek.
2. Interpreter bazlÄ± sanallaÅŸtÄ±rmalarla kodun sÃ¼rekli olarak deÄŸiÅŸken tutulmasÄ±.

1.2.0.8 8. Ã‡eÅŸitlendirme (Diversity) Teorik AÃ§Ä±klama: Ã‡eÅŸitlendirme, her bir programÄ±n farklÄ± bir versiyonunu oluÅŸturarak, kodun sabit bir yapÄ±da olmamasÄ±nÄ± saÄŸlar. Bu, virÃ¼slerin veya kÃ¶tÃ¼ niyetli yazÄ±lÄ±mlarÄ±n kodu analiz etmesini zorlaÅŸtÄ±rÄ±r.

Uygulama Ã–rnekleri:

1. AynÄ± iÅŸlevi yerine getiren ancak farklÄ± gÃ¶rÃ¼nÃ¼mlerdeki kod yapÄ±larÄ± oluÅŸturma.
2. Her kod versiyonunda kÃ¼Ã§Ã¼k yapÄ±sal deÄŸiÅŸiklikler yaparak kodun analiz edilmesini zorlaÅŸtÄ±rma.

1.2.0.9 9. Åžifreleme ve SayÄ±sal DÃ¶nÃ¼ÅŸÃ¼mler (Encoding and Transforming) Teorik AÃ§Ä±klama: Kodun bazÄ± bÃ¶lÃ¼mleri, Ã¶zel ÅŸifreleme algoritmalarÄ±yla gizlenebilir. Bu, kodun analizini zorlaÅŸtÄ±ran baÅŸka bir karartma tekniÄŸidir. Ã–zellikle sayÄ±lar Ã¼zerinde ÅŸifreleme ve dÃ¶nÃ¼ÅŸÃ¼mler uygulanabilir.

Uygulama Ã–rnekleri:

1. Kod iÃ§inde kullanÄ±lan sayÄ±larÄ± ÅŸifreleyerek bu sayÄ±larÄ±n analizini zorlaÅŸtÄ±rma.
2. ÅžifrelenmiÅŸ sayÄ±larÄ±n Ã§Ã¶zÃ¼mlerini analiz ederek orijinal deÄŸerleri geri dÃ¶ndÃ¼rme.

1.2.0.10 10. Opaque Ö°fadeler ve Dinamik Karartma (Opaque Expressions and Dynamic Obfuscation) Teorik AÃ§Ä±klama: Opaque ifadeler, kodun belirli kÄ±sÄ±mlarÄ±nÄ±n karmaÅŸÄ±k koÅŸullar altÄ±nda deÄŸerlendirilmesini saÄŸlar. Dinamik karartma, kodun Ã§alÄ±ÅŸma zamanÄ±nda sÃ¼rekli olarak dÃ¶nÃ¼ÅŸtÃ¼rÃ¼lmesi ve deÄŸiÅŸken tutulmasÄ±nÄ± iÃ§erir.

Uygulama Ã–rnekleri:

1. Kodun Ã§alÄ±ÅŸtÄ±ÄŸÄ± sÄ±rada sÃ¼rekli olarak dÃ¶nÃ¼ÅŸÃ¼mler uygulayarak analiz edilmesini zorlaÅŸtÄ±rmak.
2. Ã‡alÄ±ÅŸma zamanÄ±nda kodu yeniden yapÄ±landÄ±rarak sabit kalmasÄ±nÄ± engellemek.

$7.Hafta - Sonu$