CE407 GÃ¼venli Programlama Hafta-2

GeliÅŸtirme OrtamÄ± GÃ¼venliÄŸi ve YazÄ±lÄ±m GeliÅŸtirme SÃ¼reÃ§leri

Yazar: Dr. Ã–Ğr. Üyesi UÄŸur CORUH

Contents

List of Figures

List of Tables

1 CE407 GÃ¼venli Programlama

1.1 Hafta-2

1.1.0.1 GeliÅŸtirme OrtamÄ± GÃ¼venliÄŸi ve YazÄ±lÄ±m GeliÅŸtirme SÃ¼reÃ§leri

Ö°ndir PDF[1],DOCX[2], SLIDE[3], PPTX[4]

1.1.1 Outline

- GeliÅŸtirme OrtamÄ± GÃ¼venliÄŸi ve YazÄ±lÄ±m GeliÅŸtirme SÃ¼reÃ§leri
- YazÄ±lÄ±m GeliÅŸtirme SÃ¼reci
  - YazÄ±lÄ±m GeliÅŸtirme AkÄ±ÅŸÄ±
  - KonfigÃ¼rasyon Sabitleme
  - DeÄŸiÅŸiklik BaÅŸlatma ve SÄ±nÄ±flandÄ±rma
  - DeÄŸiÅŸikliÄŸi Onaylama ve YayÄ±nlama
- YazÄ±lÄ±m GeliÅŸtirme OrtamlarÄ±
  - GeliÅŸtirme OrtamÄ± GÃ¼venliÄŸi
  - SÃ¼rÃ¼m Kontrol Sistemleri
  - Kaynak Kod Sunucu GÃ¼venliÄŸi
  - Sunucu OdasÄ± ve GeliÅŸtirme BilgisayarlarÄ± GÃ¼venliÄŸi

[1] pandoc_ce407-week-2.tr_doc.pdf
[2] pandoc_ce407-week-2.tr_word.docx
[3] ce407-week-2.tr_slide.pdf
[4] ce407-week-2.tr_slide.pptx

1.1.2 Hafta-13: Tigress ve Ã‡eÅŸitlilik Teknikleri

Bu hafta, kodun analiz edilmesini zorlaÅŸtÄ±ran ve programÄ± saldÄ±rÄ±lara karÅŸÄ± daha direnÃ§li hale getiren Ã§eÅŸitlilik (diversification) tekniklerini ve Tigress gibi obfuscation araÃ§larÄ±nÄ± inceleyeceÄŸiz. Bu teknikler, programÄ±n Ã§alÄ±ÅŸtÄ±ÄŸÄ± her seferinde farklÄ±laÅŸmasÄ±nÄ± saÄŸlar, bÃ¶ylece saldÄ±rganlarÄ±n aynÄ± yÃ¶ntemlerle programÄ± analiz etmelerini zorlaÅŸtÄ±rÄ±r.

1.1.2.1 1. Tigress Ã‡eÅŸitlilik (Diversity) Teorik AÃ§Ä±klama: Tigress, bir programÄ± farklÄ± ÅŸekillerde dÃ¶nÃ¼ÅŸtÃ¼rerek, saldÄ±rÄ±lara karÅŸÄ± direnÃ§li hale getiren gÃ¼Ã§lÃ¼ bir obfuscation aracÄ±dÄ±r. Bir programÄ±n her Ã§Ä±ktÄ±sÄ± benzersiz bir yorumlayÄ±cÄ± (interpreter) oluÅŸturur. Bu, programÄ±n davranÄ±ÅŸÄ±nÄ± rastgeleleÅŸtirir ve analiz edilmesini zorlaÅŸtÄ±rÄ±r.

- **Tigressâ€™te KullanÄ±lan YÃ¶ntemler:**
  - **Instruction Dispatch TÃ¼rleri:**
    - Switch, direkt, endirekt, Ã§aÄŸrÄ± (call), if-else, lineer, binary, interpolasyon.
  - **Operand TÃ¼rleri:**
    - YÄ±ÄŸÄ±n (stack), registerlar.
  - **RastgeleleÅŸtirilen OperatÃ¶rler:**
    - FarklÄ± operandlar ve operator kombinasyonlarÄ± kullanarak kodun karmaÅŸÄ±klaÅŸtÄ±rÄ±lmasÄ±.
  - **Ã‡eÅŸitli DÃ¶nÃ¼ÅŸÃ¼mler:**
    - **Code Flattening:** ProgramÄ±n akÄ±ÅŸ kontrolÃ¼nÃ¼n dÃ¼zleÅŸtirilmesi.
    - **Merge/Split Fonksiyonlar:** BirleÅŸtirilen ya da bÃ¶lÃ¼nen fonksiyonlar.
    - **Opaque Predicates:** Kodda gizli ve deÄŸiÅŸtirilemeyen koÅŸul ifadeleri ekleme.

Uygulama Ã–rneÄŸi:

1. Tigress aracÄ±nÄ± kullanarak bir programÄ±n nasÄ±l benzersiz bir yorumlayÄ±cÄ±ya dÃ¶nÃ¼ÅŸtÃ¼rÃ¼ldÃ¼ÄŸÃ¼nÃ¼ gÃ¶rmek iÃ§in aÅŸaÄŸÄ±daki komutlarÄ± kullanÄ±n:

```
tigress --Transform=Virtualize --Functions=fib --VirtualizeDispatch=switch --out=v1.c test1.c
gcc -o v1 v1.c
```

Bu iÅŸlem, fib fonksiyonunu switch tabanlÄ± bir sanal makineye dÃ¶nÃ¼ÅŸtÃ¼rÃ¼r.

1.1.2.2 2. Kodda Ã‡eÅŸitlilik SaÄŸlama Teorik AÃ§Ä±klama: Ã‡eÅŸitlilik, kodun analizini zorlaÅŸtÄ±rmak amacÄ±yla farklÄ± yÃ¶ntemlerle rastgeleleÅŸtirilmesini iÃ§erir. Bu yÃ¶ntemler, bir saldÄ±rganÄ±n programÄ± tersine mÃ¼hendislikle Ã§Ã¶zmesini zorlaÅŸtÄ±rÄ±r. Tigress ile bir program her Ã§alÄ±ÅŸtÄ±rÄ±ldÄ±ÄŸÄ±nda benzersiz bir sanal makine oluÅŸturulabilir.

- **Kodda Ã‡eÅŸitlilik SaÄŸlayan Teknikler:**
  - **Flattening (DÃ¼zleÅŸtirme):** Programdaki tÃ¼m kontrol akÄ±ÅŸlarÄ±nÄ± bir dÃ¶ngÃ¼ iÃ§ine yerleÅŸtirerek karÄ±ÅŸtÄ±rma.
  - **Fonksiyon BirleÅŸtirme:** Birden fazla fonksiyonun birleÅŸtirilmesi.
  - **Rastgele SayÄ±larla Kodda Ã‡eÅŸitlilik SaÄŸlama:** Rastgele sayÄ±lar kullanÄ±larak fonksiyonlarÄ±n ve operandlarÄ±n karmaÅŸÄ±klaÅŸtÄ±rÄ±lmasÄ±.
  - **Instruction Dispatch (Talimat YÃ¶nlendirme) TÃ¼rleri:**
    - **Switch-based Dispatching:** Switch case kullanarak sanal makinelerin talimatlarÄ±nÄ± yÃ¶nlendirme.
    - **Indirect Dispatching:** Endirekt olarak dallanma noktalarÄ±nÄ± yÃ¶nlendirme.

Uygulama Ã–rneÄŸi:

1. AÅŸaÄŸÄ±daki komutlarla farklÄ± tÃ¼rde talimat yÃ¶nlendirmeler ile programÄ± sanal makinelerle Ã§alÄ±ÅŸtÄ±rÄ±n:

```
tigress --Transform=Virtualize --Functions=fib --VirtualizeDispatch=indirect --out=v2.c test1.c
gcc -o v2 v2.c
```

1.1.2.3 3. SaldÄ±rÄ±lar ve KarÅŸÄ± SaldÄ±rÄ±lar Teorik AÃ§Ä±klama: Bir saldÄ±rgan, programÄ±n sanal talimat setini Ã§Ã¶zerek kodun nasÄ±l Ã§alÄ±ÅŸtÄ±ÄŸÄ±nÄ±

anlamaya Ã§alÄ±ÅŸabilir. Bunun iÃ§in Ã§eÅŸitli saldÄ±rÄ± yÃ¶ntemleri geliÅŸtirilmiÅŸtir, ancak Tigress bu saldÄ±rÄ±lara karÅŸÄ± bazÄ± karÅŸÄ± saldÄ±rÄ± teknikleri sunar.

- **SaldÄ±rÄ± TÃ¼rleri:**
  - **SaldÄ±rÄ± 1:** TalimatlarÄ± tersine mÃ¼hendislik yaparak yorumlama.
  - **SaldÄ±rÄ± 2:** Dinamik saldÄ±rÄ±larla, programÄ± Ã§alÄ±ÅŸtÄ±rÄ±p sanal program sayacÄ±nÄ± (PC) izleyerek talimatlarÄ± Ã§Ã¶zme.
- **KarÅŸÄ± SaldÄ±rÄ±lar:**
  - **Kompleks SemantiÄŸi Olan Talimatlar Kullanma:** TalimatlarÄ±n iÃ§eriÄŸini daha karmaÅŸÄ±k hale getirerek tersine mÃ¼hendisliÄŸi zorlaÅŸtÄ±rmak.
  - **Birden Fazla Program SayacÄ± Kullanma:** Programda birden fazla PC oluÅŸturarak, saldÄ±rganÄ±n hangi PCâ€™yi izleyeceÄŸini bulmasÄ±nÄ± zorlaÅŸtÄ±rmak.

Uygulama Ã–rneÄŸi:

1. **SaldÄ±rÄ± Senaryosu:** Bir sanal makinenin talimat setini tersine mÃ¼hendislikle Ã§Ã¶zme.
2. **KarÅŸÄ± SaldÄ±rÄ±:** Programda birden fazla sanal makine sayacÄ± kullanarak, tersine mÃ¼hendislik yapÄ±lmasÄ±nÄ± zorlaÅŸtÄ±rma.

1.1.2.4 4. Algoritmik YÃ¶ntemler ve Ã‡eÅŸitlilik SaÄŸlama Teorik AÃ§Ä±klama:

Ã‡eÅŸitlilik saÄŸlama algoritmalarÄ±, programÄ±n Ã§alÄ±ÅŸmasÄ±nÄ± karmaÅŸÄ±klaÅŸtÄ±rmak iÃ§in Ã§eÅŸitli seviyelerde uygulanabilir. Bu yÃ¶ntemler, bir saldÄ±rganÄ±n programÄ± Ã§Ã¶zme olasÄ±lÄ±ÄŸÄ±nÄ± azaltmak iÃ§in kullanÄ±lÄ±r.

- **Algoritmik YÃ¶ntemler:**
  - **Build-and-Execute:** Kodun bir kÄ±smÄ±nÄ±n Ã§alÄ±ÅŸma zamanÄ±nda oluÅŸturulmasÄ± ve Ã§alÄ±ÅŸtÄ±rÄ±lmasÄ±.
  - **Self-Modifying Code (Kendi Kendini DeÄŸiÅŸtiren Kod):** Kodu Ã§alÄ±ÅŸma zamanÄ±nda deÄŸiÅŸtiren algoritmalar.
  - **Åžifreleme (Encryption):** Kodun bir kÄ±smÄ±nÄ±n ÅŸifrelenip Ã§alÄ±ÅŸma zamanÄ±nda Ã§Ã¶zÃ¼lmesi.
  - **Kodun TaÅŸÄ±nmasÄ± (Moving Code Around):** Kodun her Ã§alÄ±ÅŸtÄ±rÄ±ldÄ±ÄŸÄ±nda farklÄ± yerlerde Ã§alÄ±ÅŸtÄ±rÄ±lmasÄ±.
- **GranÃ¼lerlik DÃ¼zeyleri:**
  - **Dosya Seviyesi (File-Level):** TÃ¼m dosyanÄ±n ÅŸifrelenmesi veya taÅŸÄ±nmasÄ±.
  - **Fonksiyon Seviyesi (Function-Level):** Belirli fonksiyonlarÄ±n dinamik olarak deÄŸiÅŸtirilmesi.
  - **Temel Blok Seviyesi (Basic Block-Level):** ProgramÄ±n temel yapÄ± taÅŸlarÄ±nÄ±n karÄ±ÅŸtÄ±rÄ±lmasÄ±.

Uygulama Ã–rneÄŸi:

1. ProgramÄ± kendi kendini deÄŸiÅŸtiren bir kodla koruma:

```
void makeCodeWritable(caddr_t first, caddr_t last) {
  // Kodu Ã§alÄ±ÅŸmadan Ã¶nce deÄŸiÅŸtir.
}
```

1.1.2.5 5. Kendini DeÄŸiÅŸtiren Kod (Self-Modifying Code) Teorik AÃ§Ä±klama:

Kendi kendini deÄŸiÅŸtiren kodlar, programÄ±n Ã§alÄ±ÅŸma zamanÄ±nda kendini deÄŸiÅŸtirmesine izin verir. Bu yÃ¶ntem, bir saldÄ±rganÄ±n kodu Ã§Ã¶zmesini zorlaÅŸtÄ±rmak iÃ§in kullanÄ±lÄ±r.

- **Kanzaki AlgoritmasÄ±:** GerÃ§ek talimatlarÄ± sahte talimatlarla deÄŸiÅŸtirir ve sahte talimatlarÄ± Ã§alÄ±ÅŸtÄ±rmadan Ã¶nce gerÃ§ek talimatlarla deÄŸiÅŸtirir.
- **Madou AlgoritmasÄ±:** Dinamik olarak fonksiyonlarÄ± birleÅŸtirir ve kodun sÃ¼rekli deÄŸiÅŸmesini saÄŸlar.

Uygulama Ã–rneÄŸi:

1. Madouâ€™nun dinamik kod birleÅŸtirme algoritmasÄ±nÄ± kullanarak programÄ± koruma:

```
gcc -o v5 v5.c
```

1.1.2.6 SonuÃ§ Bu hafta, Ã§eÅŸitlilik saÄŸlama ve kendini deÄŸiÅŸtiren kod gibi ileri dÃ¼zey kod obfuscation tekniklerini Ã¶ÄŸrendik. Bu teknikler, programlarÄ±n saldÄ±rÄ±lara karÅŸÄ± daha direnÃ§li hale getirilmesini saÄŸlar ve saldÄ±rganlarÄ±n kodu Ã§Ã¶zmesini zorlaÅŸtÄ±rÄ±r. Tigress gibi araÃ§lar, kodu rastgeleleÅŸtirerek her seferinde farklÄ± bir yapÄ± oluÅŸturur, bu da kodun analizi ve tersine mÃ¼hendislik yapÄ±lmasÄ±nÄ± daha zor hale getirir.

$$End - Of - Week - 1$$