CEN429 GÃ¼venli Programlama Hafta-5

Native C/C++ iÃ§in RASP Teknikleri

Yazar: Dr. Ã–Ÿr. Ãœyesi UÄŸur CORUH

İçindekiler

Şekil Listesi

Tablo Listesi

1 CEN429 GÃ¼venli Programlama

1.1 Hafta-5

1.1.0.1 Native C/C++ iÃ§in RASP Teknikleri Ä°ndir

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

1.1.1 Outline

- RASP (Ã‡alÄ±ÅŸma ZamanÄ± Uygulama KorumasÄ±) Nedir?
- Native C/C++ Ä°Ã§in RASP Teknikleri
- Caller APK Hash DoÄŸrulama
- Root Tespiti ve LD Preload KorumasÄ±

1.2 Hafta-5: RASP (Runtime Application Self-Protection) Native C/C++ TarafÄ±

Runtime Application Self-Protection (RASP), uygulamalarÄ±n Ã§alÄ±ÅŸma zamanÄ±nda kendi gÃ¼venliklerini saÄŸlamalarÄ±nÄ± mÃ¼mkÃ¼n kÄ±lan bir gÃ¼venlik yaklaÅŸÄ±mÄ±dÄ±r. Native C/C++ uygulamalarÄ±nda, RASP kullanarak Ã§eÅŸitli gÃ¼venlik kontrolleri gerÃ§ekleÅŸtirilebilir. Bu ders kapsamÄ±nda RASP teknikleri detaylÄ±ca aÃ§Ä±klanacak ve uygulama Ã¶rnekleriyle pekiÅŸtirilecektir.

[1] pandoc_cen429-week-5.tr_doc.pdf
[2] pandoc_cen429-week-5.tr_word.docx
[3] cen429-week-5.tr_slide.pdf
[4] cen429-week-5.tr_slide.pptx

1.2.0.1 1. Ã‡alÄ±ÅŸma ZamanÄ±nda Kod BloklarÄ±nÄ±n Checksum DoÄŸrulamasÄ± (Runtime CodeBlock Checksum Verification) Teorik AÃ§Ä±klama:

Ã‡alÄ±ÅŸma zamanÄ±nda belirli kod bloklarÄ±nÄ±n hash veya checksum deÄŸerleri doÄŸrulanarak, kodun deÄŸiÅŸtirilip deÄŸiÅŸtirilmediÄŸi tespit edilir. Bu yÃ¶ntem, kod manipÃ¼lasyonlarÄ±na ve kÃ¶tÃ¼ niyetli mÃ¼dahalelere karÅŸÄ± bir koruma saÄŸlar.

Uygulama Ã–rnekleri:

1. Herhangi bir kod bloÄŸunun checksum deÄŸerini hesaplama ve Ã§alÄ±ÅŸma sÄ±rasÄ±nda bu deÄŸeri karÅŸÄ±laÅŸtÄ±rma.
2. DeÄŸiÅŸiklik tespit edildiÄŸinde programÄ±n kapanmasÄ± veya hatalÄ± bir sonuÃ§ Ã¼retmesi.
3. Ã–nemli fonksiyonlarÄ±n ve kritik kod parÃ§alarÄ±nÄ±n checksum doÄŸrulamasÄ± ile korunmasÄ±.

1.2.0.2 2. Caller APK Hash ve Ä°mza DoÄŸrulamasÄ± (Caller APK Hash Verification & Signature Verification) Teorik AÃ§Ä±klama:

APK dosyalarÄ±nÄ±n hash ve imza bilgileri doÄŸrulanarak, uygulamanÄ±n yalnÄ±zca gÃ¼venilir ve imzalanmÄ±ÅŸ APK'lar tarafÄ±ndan Ã§aÄŸrÄ±lmasÄ± saÄŸlanÄ±r. Bu sayede, uygulamanÄ±n deÄŸiÅŸtirilmiÅŸ veya sahte APK'lar tarafÄ±ndan Ã§alÄ±ÅŸtÄ±rÄ±lmasÄ± engellenir.

Uygulama Ã–rnekleri:

1. APK dosyasÄ±nÄ±n hash deÄŸerini Ã§alÄ±ÅŸma sÄ±rasÄ±nda doÄŸrulama.
2. APK'nÄ±n imza bilgisini kontrol ederek yalnÄ±zca orijinal imzalanmÄ±ÅŸ APK'larÄ±n Ã§alÄ±ÅŸmasÄ±na izin verme.
3. Hash ve imza deÄŸerlerinin saklanmasÄ± ve dinamik doÄŸrulama iÅŸlemleri.

1.2.0.3 3. Rooted Cihaz Tespiti (Rooted Device Detection) Teorik AÃ§Ä±klama:

Root yetkisine sahip cihazlar, gÃ¼venlik riskleri oluÅŸturabilir. Rooted cihazlarÄ±n tespit edilmesi, bu cihazlarda uygulamanÄ±n Ã§alÄ±ÅŸmasÄ±nÄ±n engellenmesini saÄŸlar.

Root Tespit YÃ¶ntemleri:

1. **/dev/kmem DosyasÄ±:** Sistemde bu dosyanÄ±n varlÄ±ÄŸÄ± kontrol edilir. Varsa, sistemde syscall table hook ediliyor olabilir ve cihaz root yetkisine sahip olabilir.
2. **/proc/kallsyms DosyasÄ±:** sys_call_table ve compat_sys_call_table adreslerinin boÅŸ olup olmadÄ±ÄŸÄ±nÄ± kontrol etme.
3. **/default.prop ve /system/build.prop DosyalarÄ±:** Bu dosyalar okunabiliyorsa cihaz rootlanmÄ±ÅŸ olabilir.
4. **DiÄŸer Root Tespit YÃ¶ntemleri:**
   - Superuser.apk dosyasÄ±nÄ±n varlÄ±ÄŸÄ±.
   - 27047 portuna baÄŸlanma testi ile frida serverâ€™Ä±n aranmasÄ±.

Uygulama Ã–rnekleri:

1. Belirtilen dosyalarÄ±n varlÄ±ÄŸÄ±nÄ± kontrol ederek root tespiti yapma.
2. Frida gibi araÃ§larÄ±n varlÄ±ÄŸÄ±nÄ± test etme ve tespit etme.
3. Root edilmiÅŸ cihazlarda uygulamanÄ±n Ã§alÄ±ÅŸmasÄ±nÄ± engelleme.

1.2.0.4 4. Ä°leri Seviye LD Preload SaldÄ±rÄ± Tespiti (Advanced LD Preload Attack Detection) Teorik AÃ§Ä±klama:

LD_PRELOAD, dinamik olarak yÃ¼klenen kÃ¼tÃ¼phaneleri manipÃ¼le etmek iÃ§in kullanÄ±lan bir yÃ¶ntemdir. Bu teknik, kÃ¶tÃ¼ amaÃ§lÄ± yazÄ±lÄ±mlar tarafÄ±ndan kullanÄ±lan bir saldÄ±rÄ± vektÃ¶rÃ¼dÃ¼r. LD_PRELOAD saldÄ±rÄ±larÄ±nÄ±n tespit edilmesi, uygulamanÄ±n gÃ¼venliÄŸini artÄ±rÄ±r.

Uygulama Ã–rnekleri:

1. Ã‡alÄ±ÅŸma zamanÄ±nda LD_PRELOAD ortam deÄŸiÅŸkenlerinin kontrol edilmesi.
2. LD_PRELOAD saldÄ±rÄ±larÄ±nÄ±n tespiti iÃ§in Ã¶zel algoritmalarÄ±n kullanÄ±lmasÄ±.
3. Tespit edilen saldÄ±rÄ±lara karÅŸÄ± uygulamanÄ±n kendini korumaya almasÄ±.

1.2.0.5 5. GDB, Tracers ve EmÃ¼latÃ¶r Tespiti (GDB, Tracers, and Emulator Detection) Teorik AÃ§Ä±klama:

GDB gibi hata ayÄ±klama araÃ§larÄ±nÄ±n, izleyici (tracer) ve emÃ¼latÃ¶rlerin tespit edilmesi, saldÄ±rganlarÄ±n uygulamayÄ± analiz etmelerini ve deÄŸiÅŸtirmelerini engeller.

Uygulama Ã–rnekleri:

1. GDB ortamÄ±nÄ±n tespit edilmesi ve uygulamanÄ±n bu ortamda Ã§alÄ±ÅŸmamasÄ±nÄ± saÄŸlama.
2. ltrace, strace gibi izleyicilerin kullanÄ±mÄ±nÄ± algÄ±lama ve engelleme.
3. EmÃ¼latÃ¶r ortamÄ±nda Ã§alÄ±ÅŸÄ±rken uygulamanÄ±n kapanmasÄ±nÄ± veya farklÄ± bir davranÄ±ÅŸ sergilemesini saÄŸlama.

1.2.0.6 6. Debugger Eklentisi Tespiti (Debugger Attachment Check) Teorik AÃ§Ä±klama:

UygulamanÄ±n bir hata ayÄ±klayÄ±cÄ±ya (debugger) eklenip eklenmediÄŸi tespit edilerek, kÃ¶tÃ¼ niyetli kiÅŸilerin uygulamayÄ± analiz etmesi engellenebilir.

Uygulama Ã–rnekleri:

1. Debugger eklentisini algÄ±layan kod parÃ§alarÄ±nÄ±n uygulamaya eklenmesi.
2. Debugger tespit edildiÄŸinde uygulamanÄ±n Ã§alÄ±ÅŸmasÄ±nÄ± durdurma veya farklÄ± bir iÅŸlev sergilemesini saÄŸlama.
3. Anti-debugging teknikleri ile uygulamanÄ±n gÃ¼venliÄŸini artÄ±rma.

1.2.0.7 7. Bellek KorumasÄ± (Memory Protection) Teorik AÃ§Ä±klama:

Bellek koruma teknikleri, bellek eriÅŸimlerinin kontrol edilmesini saÄŸlar. Bellek Ã¼zerinde yapÄ±lan manipÃ¼lasyonlara karÅŸÄ± koruma saÄŸlar. Clang'Ä±n SafeStack Ã¶zelliÄŸi, bellek eriÅŸimlerini izlenebilir hale getirir.

Uygulama Ã–rnekleri:

1. SafeStack kullanarak bellek koruma iÅŸlemlerinin devreye sokulmasÄ±.
2. Bellek Ã¼zerinde yapÄ±lan her tÃ¼rlÃ¼ manipÃ¼lasyonun tespit edilmesi.
3. Bellek koruma mekanizmalarÄ± ile uygulamanÄ±n gÃ¼venliÄŸini artÄ±rma.

1.2.0.8 8. DiÄŸer RASP Teknikleri

1. **LD Preload Custom Envoriment Detection:** Ã–zelleÅŸtirilmiÅŞ LD_PRELOAD ortam deÄŸiÅŞkenlerinin tespiti.
2. **Tamper Device Detection:** Uygulama cihazÄ±nÄ±n deÄŸiÅŞtirilip deÄŸiÅŞtirilmediÄŸinin kontrol edilmesi.
3. **Control Flow Counter Checking:** Kontrol akÄ±ÅŞÄ±nÄ± izleyen sayaÃ§lar ile kodun manipÃ¼le edilip edilmediÄŸinin tespiti.
4. **Device Binding:** UygulamanÄ±n belirli bir cihaza baÄŸlÄ± olarak Ã§alÄ±ÅŞmasÄ±nÄ± saÄŸlama.
5. **Version Binding:** UygulamanÄ±n belirli bir versiyonda Ã§alÄ±ÅŞtÄ±ÄŞÄ±ndan emin olma.

$5.Hafta - Sonu$