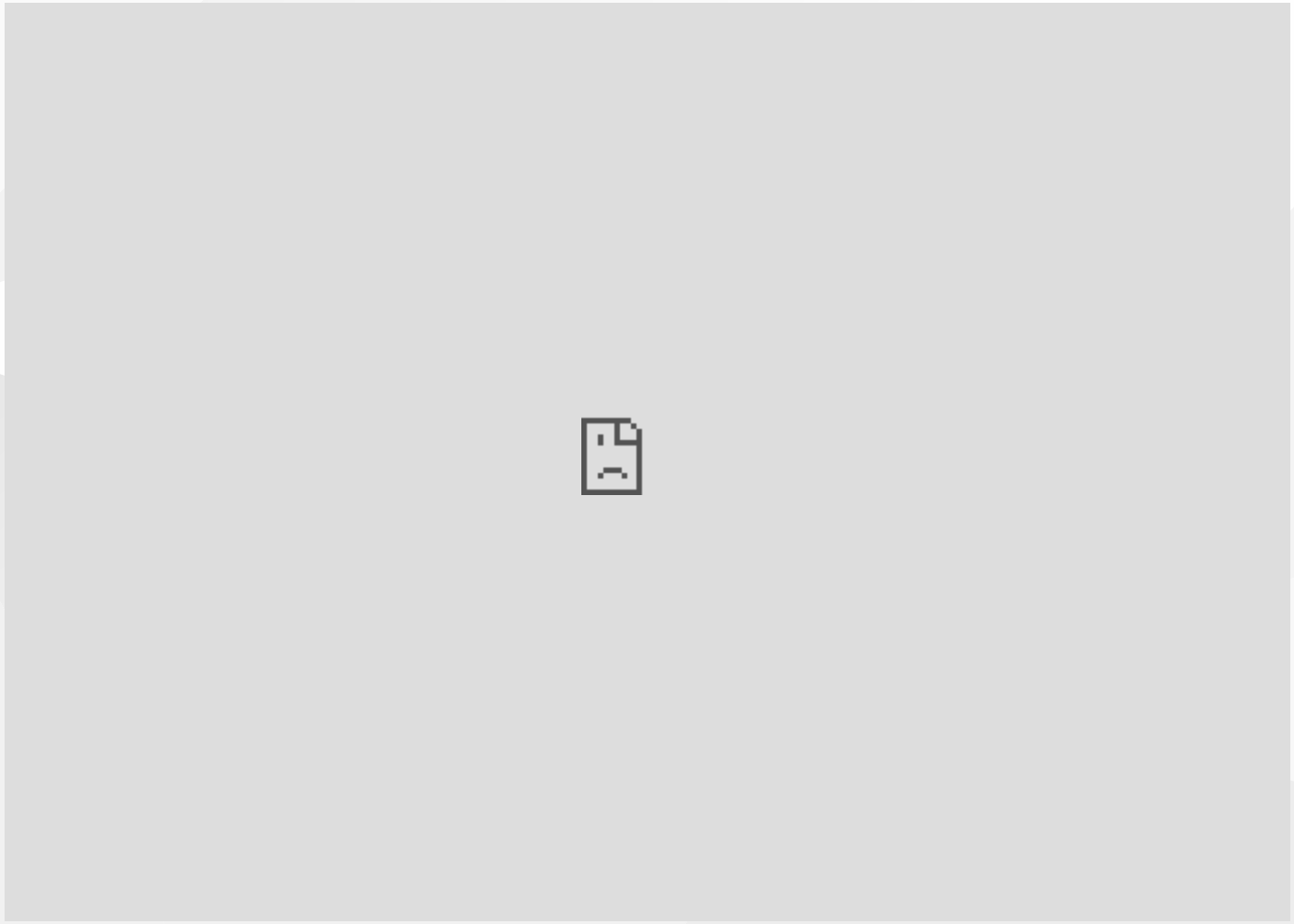
CEN429 Güvenli Programlama

Hafta-6

Java için RASP Teknikleri

İndir

- PDF
- DOC
- SLIDE
- PPTX



Outline

- RASP (Çalışma Zamanı Uygulama Koruması) Nedir?
- Java İçin RASP Teknikleri
- Emülatör, Root ve Debug Modu Tespiti
- Güvenlik Kütüphaneleri ve SSL Pinning

Hafta-6: RASP (Runtime Application Self-Protection) Java Tarafı

Java uygulamalarında RASP (Runtime Application Self-Protection), uygulamaların çalışma zamanında güvenliklerini sağlamak için kullanılan tekniklerden oluşur. Bu hafta, Java tabanlı uygulamalar için RASP stratejilerini inceleyeceğiz. Uygulamalar, özellikle mobil uygulamalar, çalışma zamanında çeşitli tehditlere karşı kendilerini koruyabilmelidir. Aşağıdaki başlıklar, Java tarafında RASP için kullanılan teknikleri kapsamaktadır.

1. Emülatör Tespiti (Emulator Detection)

Teorik Açıklama: Emülatörler, saldırganların uygulamayı analiz etmeleri ve zayıf noktaları keşfetmeleri için kullanabilecekleri araçlardır. Emülatör tespiti, uygulamanın bir emülatör ortamında çalışıp çalışmadığını anlamasına olanak tanır. Qemu gibi popüler emülatörler için özel tespit mekanizmaları uygulanabilir.

Kaynak ve Uygulama:

- Qemu ARM Emülatör Tespiti için kullanılan bir örnek: Anti Emulator for Qemu ARM
- Emülatör ortamını algılama ve çalışma sürecinde uygulamanın işlevini değiştirme.

2. Hata Ayıklama Modu Tespiti (Debug Mode Detection)

Teorik Açıklama: Bir uygulamanın hata ayıklama (debug) modunda çalışması, kötü niyetli kişilerin uygulamayı analiz etmeleri için bir fırsat sağlar. Uygulamanın hata ayıklama modunda olup olmadığını tespit etmek, bu modda çalışmasını engelleyerek güvenliği artırır.

Uygulama Örnekleri:

1. Uygulamanın çalışma zamanında hata ayıklama modunda olup olmadığını kontrol eden kod parçacıkları eklemek.
2. Hata ayıklama modunda olduğunda uygulamanın çalışmasını sonlandırmak veya farklı bir işlev sergilemesini sağlamak.

3. Debugger Bağlantısı Tespiti (Debugger Attach Detection)

Teorik Açıklama: Hata ayıklayıcıların (debugger) uygulamaya bağlanması, uygulamanın izlenmesine ve analiz edilmesine yol açar. Debugger tespiti, uygulamanın çalışma sırasında bir hata ayıklayıcıya bağlanıp bağlanmadığını kontrol eder ve buna göre hareket eder.

Uygulama Örnekleri:

1. Debugger tespit edildiğinde uygulamanın kapanmasını veya işlev değiştirmesini sağlama.
2. Hata ayıklayıcıya bağlantıyı algılayan güvenlik mekanizmaları eklemek.

4. RootBeer Implementasyonu (RootBeer Implementation)

Teorik Açıklama: RootBeer, Android cihazlarının root olup olmadığını kontrol eden bir kütüphanedir. Root edilmiş cihazlar, uygulamanın güvenliğini tehlikeye atabilir. RootBeer kullanarak, root edilmiş cihazların tespiti yapılabilir.

Uygulama Örnekleri:

1. RootBeer kullanarak cihazın root olup olmadığını tespit etme.
2. Root edilmiş cihazlarda uygulamanın çalışmasını engelleme veya kısıtlı işlev sağlama.

5. AndroidSecurityManager ile Root Tespiti (AndroidSecurityManager Rooted Device Check)

Teorik Açıklama: AndroidSecurityManager, Android cihazlarının güvenlik durumu hakkında bilgi sağlayan bir güvenlik yöneticisidir. Root edilmiş cihazları tespit ederek uygulamanın bu cihazlarda çalışmamasını sağlar.

Uygulama Örnekleri:

1. AndroidSecurityManager kullanarak root kontrolü gerçekleştirme.
2. Root edilmiş cihazlarda belirli özellikleri devre dışı bırakma.

6. SafetyNet Implementasyonu (SafetyNet Implementation)

Teorik Açıklama: Google SafetyNet, cihazın güvenlik durumunu değerlendirmek için kullanılan bir API'dir. Uygulamalar, SafetyNet ile cihazın güvenlik bütünlüğünü kontrol edebilir ve güvenlik ihlalleri tespit edildiğinde tepki verebilir.

Uygulama Örnekleri:

1. SafetyNet API'yi kullanarak cihazın güvenlik bütünlüğünü kontrol etmek.
2. Güvenlik ihlalleri tespit edildiğinde uygulamanın davranışını değiştirmek veya sonlandırmak.

7. Kullanılan Native Kütüphane Checksum Kontrolü (Used Native Library Checksum Control)

Teorik Açıklama: Uygulamanın kullandığı native kütüphanelerin checksum değerlerini kontrol etmek, bu kütüphanelerin değiştirilip değiştirilmediğini anlamamızı sağlar. Bu, uygulamanın güvenliğini korumanın önemli bir yoludur.

Uygulama Örnekleri:

1. Çalışma zamanında kullanılan kütüphanelerin checksum değerlerini kontrol etme.
2. Kütüphane üzerinde bir değişiklik tespit edilirse uygulamanın çalışmasını sonlandırma veya işlev değiştirme.

8. Tamper Cihaz Tespiti (Tamper Device Detection)

Teorik Açıklama: Cihazın veya uygulamanın manipüle edilip edilmediğini kontrol etmek, uygulamayı güvenlik ihlallerine karşı korur. Tamper tespiti ile cihaz veya uygulama üzerinde yapılmış herhangi bir değişikliği algılayabilirsiniz.

Uygulama Örnekleri:

1. Cihaz veya uygulamanın tamper edilmiş olup olmadığını tespit etme.
2. Tamper tespit edildiğinde uygulamanın çalışmasını durdurma veya kısıtlama.

9. SSL Pinning ve WebView SSL Pinning (SSL Pinning and Webview SSL Pinning)

Teorik Açıklama: SSL Pinning, uygulamanın belirli bir sunucuya güvenli şekilde bağlanmasını sağlamak için kullanılır. WebView üzerinde SSL pinning uygulamak, kullanıcıların sahte sunucularla bağlantı kurmasını engeller.

Uygulama Örnekleri:

1. WebView'da SSL pinning uygulayarak sunucunun kimliğini doğrulamak.
2. Yanlış sunucularla bağlantı kurulduğunda bağlantıyı kesmek.

10. Sunucu Sertifikası Kontrolü (Server Certificate Check)

Teorik Açıklama: Uygulamanın bir sunucuya bağlanırken sunucu sertifikasının doğruluğunu kontrol etmesi, sahte sunucularla bağlantı kurmayı engeller. Bu, man-in-the-middle saldırılarına karşı önemli bir koruma sağlar.

Uygulama Örnekleri:

1. Sunucu sertifikasının doğruluğunu çalışma sırasında kontrol etme.
2. Yanlış sertifika tespit edildiğinde bağlantıyı kesme.

11. Cihaz ve Sürüm Bağlama (DeviceBinding & VersionBinding)

Teorik Açıklama: Cihaz bağlama, uygulamanın belirli bir cihaz üzerinde çalışmasını sağlar ve başka bir cihazda çalışmasını engeller. Sürüm bağlama ise uygulamanın belirli bir sürümde çalıştığından emin olur.

Uygulama Örnekleri:

1. Uygulamanın sadece belirli bir cihazda çalışmasını sağlayan cihaz bağlama işlemlerini gerçekleştirme.
2. Uygulamanın yalnızca belirli sürümlerde çalışmasını kontrol eden sürüm bağlama işlemleri.

12. Tüketici Doğrulaması (Consumer Verification)

Teorik Açıklama: Uygulamanın gerçek kullanıcı tarafından kullanıldığını doğrulamak, sahte kullanıcıları ve otomatik işlemleri engellemeye yardımcı olur. Bu doğrulama işlemi, tüketicinin kimliğini doğrular.

Uygulama Örnekleri:

1. Tüketici doğrulaması için güvenlik testleri ve algoritmalar kullanmak.
2. Doğrulanmamış kullanıcılar için erişim kısıtlamaları koymak.

$6. Hafta - Sonu$